9-13-2007

Search Techniques for Multi-Objective Optimization of Mixed Variable Systems having Stochastic Responses

Jennifer G. Walston

Search Techniques for Multi-Objective Optimization of Mixed-Variable Systems Having Stochastic Responses

DISSERTATION

Jennifer G. Walston, Major, USAF

AFIT/DS/ENS/07S-06

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this dissertation are those of the author and do not reflect the official policy of the United States Air Force, Department of Defense, or the U.S. Government.

AFIT/DS/ENS/07S-06

Search Techniques for Multi-Objective Optimization of Mixed-Variable Systems Having Stochastic Responses

DISSERTATION

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Jennifer G. Walston, A.S., B.S., M.S.

Major, USAF

September 2007

AFIT/DS/ENS/07S-06

Search Techniques for Multi-Objective Optimization of Mixed-Variable Systems Having Stochastic Responses

Jennifer G. Walston, A.S., B.S., M.S.

Major, USAF

Approved:

/signed/	
Dr. James W. Chrissis (Chairman)	date
/signed/	
Dr. James T. Moore (Member)	date
/signed/	
Dr. Robert A. Canfield (Member)	date
/signed/	
Lt Col Mark A. Abramson (Member)	date
/signed/	
Lt Col James A. Fellows (Dean's Representative)	date

Accepted:

/signed/	
Marlin U. Thomas	Date

Dean, Graduate School of Engineering and Management

Abstract

A research approach is presented that extends the separate solution methods of stochastic and multi-objective optimization problems to one that would solve problems having both characteristics. Such problems are typically encountered when one desires to optimize systems with multiple, often competing, objectives that do not have a closed form representation and must be estimated, *e.g.*, via simulation. First, the class of mesh adaptive direct search (MADS) algorithms for nonlinearly constrained optimization is extended to mixed variable problems, and convergence to appropriately defined stationary points is proved. The resulting algorithm, MV-MADS, is then extended to stochastic problems (MVMADS-RS), via a ranking and selection procedure. Finally, a two-stage method is developed that combines the generalized pattern search/ranking and selection (MGPS-RS) algorithms for single-objective, mixed variable, stochastic problems with a multi-objective approach that makes use of interactive techniques for the specification of aspiration and reservation levels, scalarization functions, and multi-objective ranking and selection. This combination is devised specifically so as to keep the desirable convergence properties of MGPS-RS and MVMADS-RS, while extending to the multi-objective case. A convergence analysis for the general class of algorithms establishes almost sure convergence of an iteration subsequence to stationary points appropriately defined in the mixed-variable domain. Seven specific instances of the new algorithm are implemented and tested on 11 multi-objective test problems from the literature and an engineering design problem.

Acknowledgements

This accomplishment is not mine alone. I would be completely remiss if I did not thank the many people without whom I would certainly have crashed and burned. First, thanks to the U.S. National Science Foundation and the U.S. National Academy of Science for funding my studies in the young scientist summer program as well as many thanks to the great folks at the International Institute of Applied Systems Analysis for their patient help.

Next, I want to thank my parents, who gave me a rock solid foundation for a great life. You always put my interests above your own and were the shining example of parenting that I hope to follow with my own kids. Thanks for all the encouragement and for reminding me that there is more to life than graduate school and that there really is a light at the end of the tunnel. Thanks also to my brother for believing in me and for always being the proverbial big bro. Thanks also go out to my mother and father in-law for being a great help with the kids during times of emergency.

There are not enough words to thank my husband and sons for their support over the last three years. To my husband, thanks and love for his understanding and patience during this endeavor. It hasn't always been easy, but things worth doing rarely are. To my beloved sons, hugs and kisses for being such great kids through all of this. You guys are the two best sons a Mommy can have and are truly the bright spot in my day and the very best part of my life.

Finally, I would like to thank and praise my Lord and Savior Jesus Christ for giving me courage to seek a PhD, strength to finish the program, and wisdom to remember what's really important.

Jennifer G. Walston

Table of Contents

Page

List of Figures

List of Tables

Search Techniques for Multi-Objective Optimization of Mixed-Variable Systems Having Stochastic Responses

I. Introduction

1.1 Problem Setting

With the advent of advanced numerical techniques for analyzing complex engineering problems, engineers are seeking to integrate such methods into smart design processes. As a result, optimization techniques have become increasingly important in engineering design problems, *e.g.*, in the area of conceptual aircraft design [99]. However, complications arise when applying traditional optimization techniques to engineering design problems like aircraft design. Such design problems typically contain multiple, often competing, objectives [35, 77, 82, 90]. Additionally, these objectives are often subject to measurement error or must be estimated by complex simulations [35,36,81,91]. Finally, engineering design problems often contain discrete or categorical design variables [82]. Thus, multi-objective and stochastic optimization techniques, applicable to both continuous and discrete decision variables, must be considered. However, significant challenges exist for these types of optimization problems.

1.1.1 Modeling Uncertainty. Given a general constrained nonlinear optimization problem, formulated as

$$\begin{aligned} &\min F(x) \\ &\text{subject to} \\ &\qquad g_i(x) \leq 0, \quad i = 1, 2, \ldots, M, \end{aligned}$$

where $x \in \mathbb{R}^n$ represents the controllable design variables, $F : \mathbb{R}^n \to \mathbb{R}$, and the constraints $g_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, 2, \ldots, M$, define the feasible region $\Omega \subseteq \mathbb{R}^n$, a myriad

of archetypal solution methods are available to the analyst (*e.g.*, Newton's method, steepest descent, etc.) [162]. But if one or more elements of the optimization problem contain a random component, what changes to classical optimization are required in order to allow reasonably efficient optimization of this random system? The answer to this question involves the class of solution methods called *stochastic optimization* which, as the name implies, deals with systems in which one or more of the problem components—objective function(s) and/or constraints—contain a random element. Stochastic optimization is related to traditional statistical modeling, in that one seeks to model a function by repeated sampling of a population; however, in this case, the function is dependent not only on a random element, but also on controllable design variables.

A general stochastic optimization problem can be written as

$$\min F(x, \omega) \tag{1.2a}$$

subject to

$$g_i(x, \omega) \leq 0, \quad i = 1, 2, \ldots, M \tag{1.2b}$$

where $x \in \mathbb{R}^{n_1}$ and $\omega \in \mathbb{R}^{n_2}$ represent the controllable design variables and random (uncontrollable environmental) variables, respectively, and the constraint set $g_i(x, \omega)$, $i = 1, 2, \ldots, M$ defines some feasible region $\Omega \subseteq (\mathbb{R}^{n_1} \times \mathbb{R}^{n_2})$. Thus, the task becomes finding x so that Equation (1.2a) is minimized in a certain sense over all feasible x and all possible values of ω. Notions of *feasibility* and *optimality* for stochastic systems are highly dependent on the specific problem under study and must be precisely defined [48].

This research focuses on problems in which all constraints are deterministic and the objective function cannot be explicitly evaluated and must be estimated through some kind of simulation. Here *simulation* refers to a generic numerical method by which input (control) variables are used to produce an output measure of interest (response) [53, 138]. Therefore, in this *simulation-based optimization*, the observed

system response is a function of both the design variables and the random error associated with the simulation model.

For simulation-based optimization, the general stochastic objective function given in (1.2a) is typically replaced by its mathematical expectation $E[\cdot]$ [48]. With this convention and the assumption that the observed response is an unbiased approximation of the true system response, the observed response F can be represented by $F(x,\omega) = f(x) + \varepsilon_\omega(x)$ where f is the deterministic, "true" objective function, and $\varepsilon_\omega : \mathbb{R}^{n_1} \to \mathbb{R}$ is the random error function associated with the simulation (*i.e.*, the random element of F due to the uncontrollable environmental variables ω) such that $E[\varepsilon_\omega(x)] = 0$. Thus, the problem is reformulated as

$$\min F(x,\omega) \cong E[F(x)] = E[f(x) + \varepsilon_\omega(x)]$$

subject to

$$g_i(x) \leq 0, \quad i = 1, 2, \ldots, M.$$

1.1.2 Optimizing Multiple Objectives. In addition to the complexity of design optimization problems due to the stochastic element, often there exists no single criterion for choosing the *best* solution. In fact, even the notion of "best" can be unclear when multiple objectives are present. In many cases, it can be shown that improvement to one objective actually degrades the performance of another. Such objectives are called *competing objectives* and are the motivation for the study of multi-objective optimization as a separate discipline; if the objectives did not compete, they could be collapsed into a single objective and classic solution techniques would apply [1]. The multi-objective optimization problem,

$$\min E[F(x)] = E[f(x) + \varepsilon_\omega(x)] \tag{1.4a}$$

subject to

$$x \in \Omega = \{x \in \mathbb{R}^n : g_i(x) \leq 0, \quad i = 1, 2, \ldots, M\}, \tag{1.4b}$$

where $F : \mathbb{R}^n \rightarrow \mathbb{R}^J$, is that of finding a solution $x^* \in \Omega$ that optimizes the set of objectives $F = (F_1, F_2, \ldots, F_J)$ in the sense that no other point $y \in \Omega$ yields a better function value in all the objectives [83]. The point x^* is said to be *non-dominated*, *efficient* or *optimal in the Pareto sense* [54]. Further definitions associated with Pareto optimality follow.

1.1.2.1 Definitions Associated with Pareto Optimality. When optimizing over multiple objectives, the space containing the decision variables is referred to as the *decision space*, and the space containing their associated images (*i.e.*, objective function values) is referred to as the *objective space*. Two particular points in the objective space, which are often referenced in multi-objective optimization methods, are the *utopia point* (or *ideal point*) and *nadir point*. These are formally defined in Definitions 1.1.1 and 1.1.2, respectively [46], and are depicted in Figure 1.1 by points **U** and **N**, respectively.

Definition 1.1.1. *The point* $y^U = (y_1^U, \ldots, y_J^U)$ *given by* $y_k^U := \min_{x \in \Omega} F_k(x)$ *is called the* utopia *or* ideal point *of the multi-objective optimization problem* $\min_{x \in \Omega}(F_1(x), \ldots, F_J(x))$.

Definition 1.1.2. *The point* $y^N = (y_1^N, \ldots, y_J^N)$ *given by* $y_k^N := \max_{x \in \Omega} F_k(x)$ *is called the* nadir *point of the multi-objective optimization problem* $\min_{x \in \Omega}(F_1(x), \ldots, F_J(x))$.

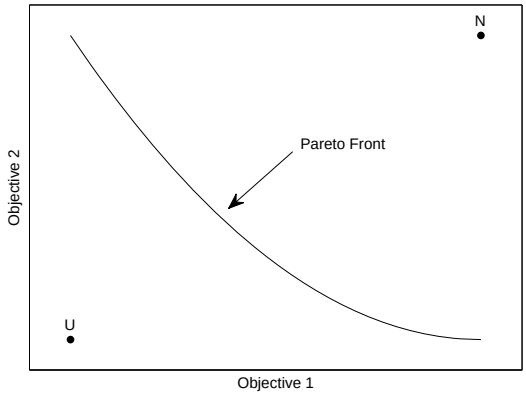


Figure 1.1: Graphical Depiction of the Objective Space (Convex Pareto Front)

There are several equivalent definitions of Pareto optimal solutions—also called *non-dominated* or *efficient* solutions—in the literature. For the purpose of this research, the definition presented as Definition 1.1.3 is used [46].

Definition 1.1.3. *A solution $x^* \in \Omega$ to the multi-objective optimization problem, given in Equations (1.4a-1.4b), is said to be Pareto optimal if there is no $x \in \Omega$ such that $F_k(x) \leq F_k(x^*)$ for all $k = 1, 2, \ldots, J$ with $F_i(x) < F_i(x^*)$ for some $i \in \{1, 2, \ldots, J\}$.*

Definition 1.1.4. *A solution $x^* \in \Omega$ to the multi-objective optimization problem, given in Equations (1.4a-1.4b), is said to be a* Pareto stationary point *or to satisfy* first-order necessary conditions for Pareto optimality *if there exists no feasible direction $d \in \mathbb{R}^n$ such that $\nabla F_k(x^*)^T d \leq 0$ for all $k = 1, 2, \ldots, J$ and $\nabla F_i(x^*)^T d < 0$ for some $i \in \{1, 2, \ldots, J\}$.*

In this research, the set of Pareto optimal solutions is referred to as the *Pareto optimal set* or simply the *Pareto set*. The image of the Pareto set is referred to as the *Pareto Frontier* or *Pareto Front*. A graphical example of a convex Pareto front is shown in Figure 1.1; a non-convex Pareto front is shown in Figure 1.2. If the Pareto set (or corresponding Pareto front) results from a solution algorithm and is not exact, it is referred to as the *approximate* (or *experimental*) *Pareto set* or *approximate* (or *experimental*) *Pareto front*, respectively.

1.1.3 Optimizing Over a Discrete Decision Space. The complexity of these problems is further increased when the decision space includes variables that are either discrete (*e.g.*, integer-valued) or categorical. Categorical variables are those which can only take on values from a predetermined list, and may not even have an ordinal relationship to each other. However, since categorical variables can be mapped easily to discrete-numeric values, these two types of variables will be grouped together and considered as a single variable type in this research.

Discrete variables are common in engineering design problems. For example, in the design of aircraft, the number of engines is integer-valued and the engine type

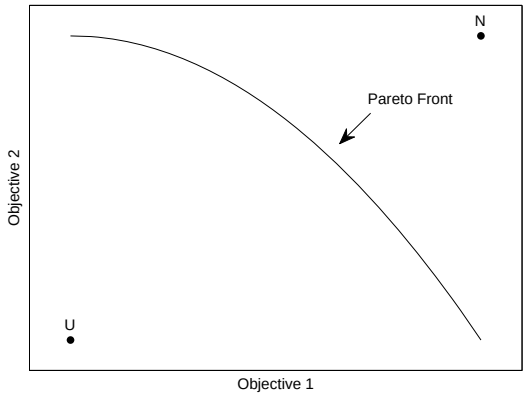


Figure 1.2: Graphical Depiction of the Objective Space (Non-Convex Pareto Front)

(turboprop, turbofan, etc.) and engine placement (wing, aft, or combination) [126] are categorical. Other discrete variables may include airfoil type, wing configuration (orientation and location of the lifting surfaces of the aircraft [8, 127]), and cabin layout (number of aisles, location of lavatories, etc. [127]). The class of optimization problems that may contain continuous, discrete-numeric, and categorical variables is known as *mixed variable programming* (MVP) [9, 138].

To model mixed variables, the decision space is partitioned into continuous and discrete domains, Ω^c and Ω^d, respectively, where the discrete variables may include categorical variables. Without loss of generality, discrete variables can be mapped to the integers, so that the discrete part of the decision space can be represented as a subset of the integers, *i.e.*, $\Omega^d \subseteq \mathbb{Z}^{n^d}$, where n^d is the dimension of the discrete space. A solution $x \in \Omega$ is denoted by $x = (x^c, x^d)$, where $x^c \in \mathbb{R}^{n^c}$ and $x^d \in \mathbb{Z}^{n^d}$.

1.1.4 Problem Formulation. The inclusion of stochastic and multi-objective elements to the classic optimization problem formulation yields the following restated

problem:

$$\min E[F(x)] = E[f(x) + \varepsilon_\omega(x)] \tag{1.5a}$$

subject to

$$g_i(x) \leq 0, \quad i = 1, 2, \ldots, M, \tag{1.5b}$$

where $x \in (\mathbb{R}^{n^c} \times \mathbb{Z}^{n^d})$ and $F : (\mathbb{R}^{n^c} \times \mathbb{Z}^{n^d}) \to \mathbb{R}^J$.

1.2 Requirements Analysis and Research Outline

Because engineering design optimization and many other practical optimization applications are generally multi-objective and may contain both stochastic elements and mixed variables, an optimization method capable of handling the following four characteristics is desired.

1. **Stochastic.** Stochastic optimization algorithms should be able to approximate the Pareto frontier when starting from an arbitrary point for problems in which function evaluations are known to contain measurement error or must be estimated, *e.g.*, via simulation. Further, such an algorithm should be provably convergent to Pareto solutions, thus guaranteeing that a representation of the frontier in a region of interest can be found. However, convergence for stochastic methods is usually specified as "almost surely" or "with probability 1" [137].
2. **Multi-Objective.** Ideally, algorithms should be capable of finding a reasonably accurate approximation of the Pareto frontier in cases for which no preference information is explicitly known or even exists. However, in many engineering design problems, some information about desired performance goals (called *aspiration levels*), as well as minimum acceptable performance levels (called *reservation levels*), may in fact exist, and can be used to determine a region of interest in which to estimate the Pareto front. This type of preference information is assumed to exist.

3. **General Purpose.** An algorithm should be applicable to a wide range of problems, with any combination of variable types. An algorithm should also be indifferent or robust to the source of the function evaluations; *i.e.*, the algorithm is able to treat the objective and constraint functions as "black boxes".

4. **Efficient.** To be practical and useful for real design problems, an algorithm should perform well with respect to the number of function evaluations required. In many design applications, such function evaluations are obtained via costly simulation runs and should therefore be used as parsimoniously as possible.

1.2.1 Problem Statement. There exists no provably convergent, general-purpose class of methods for solving multi-objective, stochastic optimization problems that apply to the mixed-variable case and are indifferent to the source of function evaluations, as well as computationally efficient algorithmic implementations of these methods.

1.2.2 Research Objectives. The purpose of this research is to develop a provably convergent, general-purpose class of methods for solving multi-objective, stochastic optimization problems that apply to the mixed-variable case and are indifferent to the source of function evaluations. Each of these requirements for the solution method presents specific challenges in the optimization process. In response to these unique challenges, a new method is introduced, which extends the applicability of generalized pattern search with ranking and selection for linearly constrained problems [138] and mesh adaptive direct search (MADS) for nonlinearly constrained problems [6] to multi-objective problems through the use of interactive specification of aspiration/reservation levels [62], scalarization functions [101], and multi-objective ranking and selection methods [85]. Specific goals of the research are:

1. Determine an appropriate amalgamation of stochastic and multi-objective methods to be extended to the multi-objective, stochastic case. Investigate tradeoffs

among the desirable characteristics of proven convergence properties, simplicity, computational efficiency, and processing time.

2. Extend previous methods to address the mixed-variable, multi-objective, stochastic class of problems. Examine and prove convergence properties of the new solution methodology.

3. Develop specific algorithmic implementations of the new solution methodology. Test the implementations on a range of appropriate test problems and evaluate computational efficiency. The following sub-objectives summarize the empirical results pursued.

   (a) Test the algorithm on a set of test problems with known solutions. In order to evaluate the algorithm's use on stochastic problems, the test problems are modified to introduce random noise into the objective function evaluations. Compare the solution quality and the computational efficiency to the published results of other solution methods. Because no methods currently exist for the mixed-variable, multi-objective, stochastic class of problems, compare the new algorithm to those used for deterministic, multi-objective problems and infer a quality evaluation of computational speed and accuracy.

   (b) Test the algorithm on an engineering design problem.

4. Study appropriate algorithm termination criteria. Develop heuristic stopping criteria based on differences in competing responses compared to variations in the responses and the practically required tolerance of the solution.

1.3 Overview

This dissertation is organized as follows. Chapter II reviews several existing solution methods for both stochastic and multi-objective problems, as well as the few that exist for problems with both characteristics. Chapter III presents specific elements of the proposed methodology, an outline of the proposed methodology, and

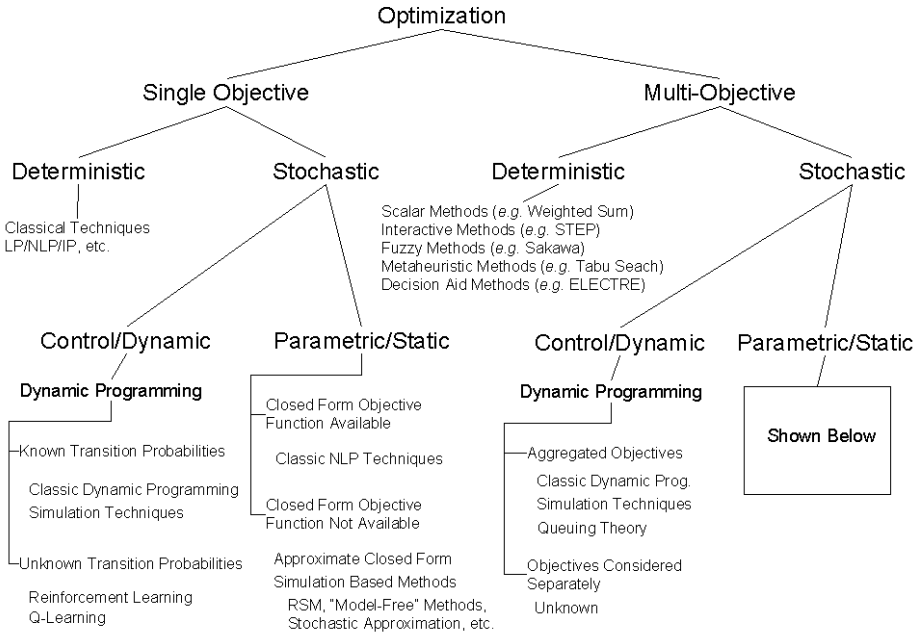
theoretical convergence results. Chapter IV provides algorithmic implementations of the solution methodology. Chapter V presents computational results of these implementations tested against multi-objective test problems with known results, as well as an engineering design optimization problem. Finally, Chapter VI summarizes the achievements of this research and suggests areas for further research.

II. Literature Review

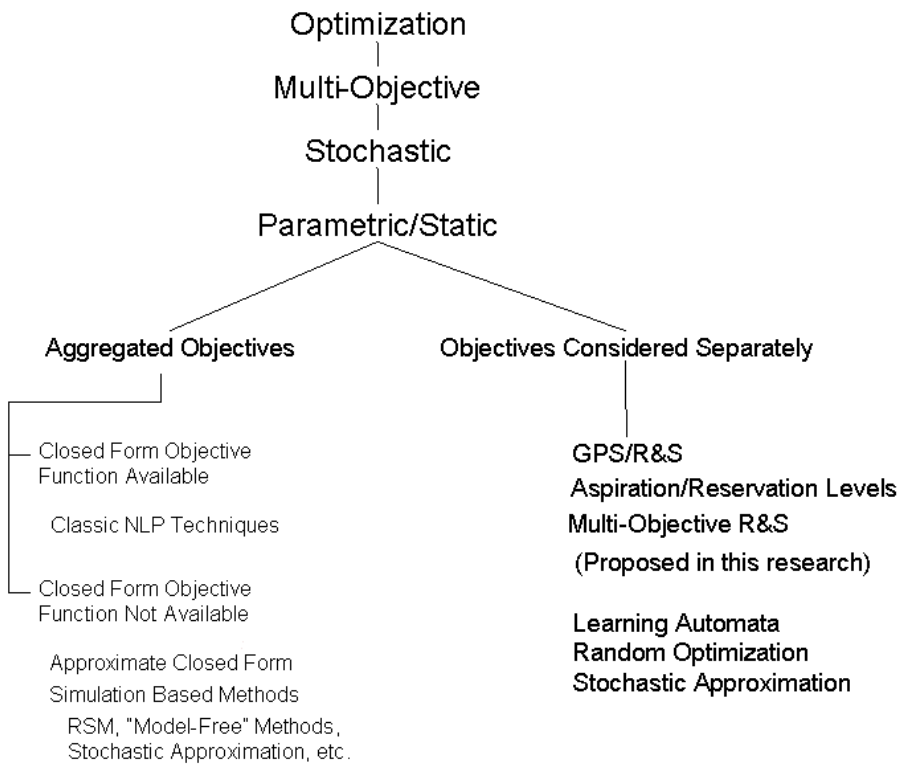
As illustrated in Figure 2.1, optimization is a diverse field with many branches. Probably the most well known and longest studied problem is that of single objective, deterministic optimization. Examples of traditional optimization techniques include Dantzig's simplex method for linear programs [162], branch and bound techniques and decomposition like Sweeney-Murphy decomposition for integer programs [142] and Benders decomposition for large-scale problems [21], nonlinear programming techniques like sequential quadratic programming [73], and heuristic methods like evolutionary algorithms. However, though applicable to many types of problems, other methods are needed for problems with more than one objective or for systems having stochastic responses. Thus, the disciplines of stochastic optimization and multi-objective optimization have been developed, but, for the most part, separately. The following sections provide an overview of stochastic and multi-objective optimization methods, as well as methods that have been used for stochastic multi-objective problems with continuous variables.

2.1 Methods for Stochastic Optimization

Stochastic optimization problems are those in which the objective or constraint functions contain a random element, either due to random noise in the function evaluations or random algorithm choices (such as the search direction) [134]. Stochastic optimization problems can be further delineated by the nature of solution required. Parametric optimization is used to find values for a set of parameters to optimize a performance measure, *e.g.*, cost, reliability, or weight. In contrast, control optimization seeks to find a set of actions to be taken in different states a system can visit in order to optimize a performance measure [61]. A classic example of this type of problem is the Markov decision problem, which is typically solved using dynamic programming. Various solution techniques for both types of problems are shown in Figure 2.1. (Note that when multiple objectives can be aggregated in some way—see Section 2.2.1—single objective control optimization techniques can be used



(a) Optimization Overview



(b) Multi-Objective Parametric Optimization of Stochastic Systems

Figure 2.1: Optimization Lineage

for multi-objective problems.) Though this research addresses the field of parametric optimization, a brief review of control optimization is presented for the sake of completeness before presenting methods of parametric optimization.

2.1.1 Solution Methods for Control Optimization.

2.1.1.1 Dynamic Programming. Developed in the late 1950's by Bellman and Howard [20, 69], dynamic programming has a considerably lower computational burden than exhaustive enumeration. It has continued to evolve over the last half century and is now considered to be the main pillar of stochastic control optimization. Dynamic programming assigns a so-called *value function* to the process in each state. Then the *Bellman equations for value iteration*—recursion equations for expected reward based on the probability of a state and the value associated with that state [155]— are used to find the optimal set of actions to move the process along the best path. Simulation can be used to estimate the value function at each state. If the control optimization problem consists of a Markov or semi-Markov [20, 69] decision process, then dynamic programming is guaranteed to converge to an optimal solution.

2.1.1.2 Reinforcement Learning. A severe drawback to dynamic programming is that the transition probabilities between states must be known in order to use the Bellman equations, which may be difficult to obtain for real-world problems. Overcoming this problem may require either simulation to approximate transition probabilities or simply using a method that does not require them. Since simulation is somewhat inefficient, the latter choice is generally preferred. Reinforcement learning is one such method [61].

Reinforcement learning was developed in the artificial intelligence community and is known as a "machine learning" technique. Though only guaranteed to find near-optimal solutions, it is still a powerful solution method because it is based on the Markov decision model. In this approach, the decision maker is viewed as a

learning agent, who receives feedback from the environment based on decisions the agent makes. Good decisions produce reinforcement by increasing the probability the agent will take that action, but bad ones result in punishment by decreasing this probability. In this way, the agent learns by trial and error in a simulated environment [61]. Specific reinforcement learning methods include adaptive critic methods [15], Q-learning [61, 151], and temporal difference learning [134].

2.1.2 Methods for Stochastic Parametric Optimization. Consider the problem [61],

$$\min f(x) = \int_{-\infty}^{\infty} [x - 5]^{-0.3} p(x) dx,$$

where p is a probability density function (pdf) of the random variable x, and the objective function f is the expected value of x. Since the pdf is known and f is known and has closed form, this problem can be solved with standard nonlinear programming techniques such as steepest descent, gradient projection methods for constrained problems, and linearization methods like Frank-Wolfe [52, 102, 107]. If the closed form of the objective function is not known or has no closed form, or if it is difficult to obtain values for the pdf, other methods must be used. Types of solution methods that estimate the value of the objective function using simulation are aptly called *simulation-based optimization*. Simulation, though not an optimization technique *per se*, can be used in conjunction with numerical optimization methods in order to solve difficult parametric optimization problems [61]. Sriver provides an excellent overview of simulation-based optimization techniques [137]. These (and additional) methods are discussed in Sections 2.1.2.1–2.1.2.3.

2.1.2.1 Response Surface Methodology. Response surface methodology (RSM), developed by Box *et al.* [26, 27, 104, 105], is one of the oldest simulation-based methods of parametric optimization. It is based on the idea that an approximate form of the objective function can be obtained by simulating the system at a finite number of carefully chosen points in the decision space. Traditionally, RSM has

used polynomial regression over the sampled points to find an approximate form of the objective function; however, alternate forms have been proposed such as kriging, neuro-response surfaces, multivariate adaptive regression splines (MARS), and radial basis functions [61, 137, 138]. A survey of such methods can be found in [75]. It is decidedly robust and often works well where other methods fail [61]; however, it only converges to the optimal value of the model, not that of the original problem.

2.1.2.2 Continuous Optimization. Continuous parametric optimization refers simply to optimization problems with continuous variables. This class of problems has two general types of solution methods: those that use the gradient of the objective function and those that do not [61, 162].

Gradient-Based Methods. Stochastic gradient-based methods include gradient descent and simultaneous perturbation. First, consider gradient descent (also called the *method of steepest descent* or *finite difference stochastic approximation*). A stochastic version of one-dimensional gradient descent for finding the root of a one-dimensional unconstrained noisy function was first introduced by Robbins and Monro in the 1950s [123]. It was extended shortly thereafter by Keifer and Wolfowitz [76] to that of using central finite differences to estimate the derivative, and later by Blum to include the multivariate case [23]. It has been the most popular and widely used method for simulation-based optimization [143].

At each iteration, the partial derivatives are estimated, the new search direction is chosen as the negative of the approximated gradient, and the algorithm moves a small step in this direction. This repeats until the estimate of the gradient is close to zero (within a tolerance).

The gradient descent method, which is defined by the iteration,

$$x^{k+1} = x^k - \mu \nabla f(x^k), \;\; k = 1, 2, \ldots, \tag{2.1}$$

is guaranteed to converge to a stationary point if the function f is convex, continuously differentiable, and the step size μ is sufficiently small [61]. When the closed form of the objective function is not known, the "sufficiently small" step size must be found by experimentation. Additionally, $2n$ function evaluations are required per iteration to estimate the gradient in Equation (2.1), and since the function evaluations are generated via simulation, multiple replications may be required per function evaluation. Thus, as the number of decision variables grows, the number of (possibly time-consuming) simulation runs grows dramatically [61, 134].

Spall [134] developed *simultaneous perturbation*, a method in which the gradient can be estimated with only two function evaluations, regardless of the number of variables. The objective function is evaluated at the current iterate and at another perturbed point a small distance from the current iterate in order to approximate partial derivatives. The key is that the perturbed point is shifted (randomly) in all dimensions at once. This method is not only efficient, but also has been shown to converge with probability 1 to a stationary point of the objective function [61].

Gradient-Free Methods. The second type of continuous optimization solution methods do not require gradients, *e.g.*, *direct search* methods [137]. These methods evolved from early efforts to optimize stochastic systems. In the 1950s, Box proposed a method for improving industrial efficiency known as evolutionary operation (EVOP) [24] in which estimation of regression models was replaced by a relatively simpler method of inspecting experimental data according to patterns in the experimental design. Later, an automated version of EVOP with updated experimental designs (simplex vice factorial) was proposed by Spendley, Hext, and Himsworth [135].

During the 1960s, several more direct search methods were developed including the search method of Hooke and Jeeves [67] and the Nelder-Mead method [109]. The Nelder-Mead method, also known as the *downhill simplex* method or *flexible polygon search*, is an extension of the method developed by Spendley, Hext, and

Himsworth [135]. These two methods are perhaps the most frequently used direct search methods and have been widely used on real-world problems with much success. At the time, they were generally regarded as heuristics because they were thought to not possess satisfactory convergence properties in high dimensions [61] or a general convergence theory in the stochastic setting [137]. More recently, Torczon proved convergence for a more general class of *pattern search* methods, which includes the direct search method of Hooke and Jeeves but not the Nelder-Mead method [146]. (Torczon also states that the simplex method of Spendley, Hext, and Himsworth can be shown to converge with some modification to the original algorithm, but only under the additional assumption that f is convex [146].) Later, McKinnon [100] showed examples in which Nelder-Mead converges to nonstationary points. These examples are for well-behaved functions and illustrate the limitations on the convergence properties of Nelder-Mead [100]. A survey of pattern search methods can be found in [80].

The Nelder-Mead method starts with a feasible set of solutions (dimension $n+1$) whose vertices define a "simplex" or "polygon". At each iteration, the worst solution is dropped from the simplex in favor of a good one. During the search, the geometry of the simplex is changed by expansion, reflection, contraction, or shrinking operations based on the relative rank of the point added at each iteration [109]. Improvements and extension to stochastic problems of this method include work by Tomick, Arnold, and Barton to determine rules for estimating the number of replications required at each simplex design point [145]; a variant of the method proposed by Barton and Ivy that reduces the risk of false convergence [16]; and work by Humphrey and Wilson that addresses sensitivity to starting values, premature termination, robustness, and computational efficiency [71, 72].

The Hooke-Jeeves pattern search method involves two types of movement. First, a search area is determined via *exploratory* moves from the starting point. A pattern of points is then defined in this area and is explored by changing each parameter in turn. The pattern is then shifted to a new region via the exploratory moves [140]. This method has been used in conjunction with two-stage ranking and selection pro-

cedures [111] as well as inside the optimization modules of the GASP and SLAM simulation languages [113, 114]. Little research was done on direct search methods from the 1960s until the introduction and convergence analysis of generalized pattern search (GPS) for unconstrained optimization problems by Torczon in the 1990s [146]. GPS is discussed in more detail in Chapter III.

2.1.2.3 Discrete Optimization. Discrete optimization problems have gaps in the domain of the objective function, so derivatives may be of little use. If the solution space is sufficiently small, then the problem can be solved by exhaustive search of the solution space. However, as problem dimension increases, exhaustive search quickly becomes impractical. Additionally, even when the solution space is relatively small, a more efficient method for comparing and ranking the best solutions is preferred. [61]

Ranking and Selection. The ranking and selection techniques of Goldsman and Nelson [60] or Hochberg and Tamhane [66] are examples more efficient methods. Ranking and selection methods are statistical methods for selecting the best solution from among a set of competing candidates. These methods are theoretically sound, well tested, and are typically used on systems with up to twenty candidate solutions, although recent work shows that they can be used on much larger problems [61]. Other examples of ranking and selection algorithms are the Rinott method and the Kim-Nelson method, a more recent, and possibly more efficient algorithm [61].

Meta-heuristics. When several hundred or thousand candidate solutions exist, ranking and selection methods cannot be used directly. They can, however, be used inside of a meta-heuristic to increase the power of the ranking and selection method (by increasing the number of alternatives that can be examined) and to make the results of the meta-heuristic more reliable. Meta-heuristics focus on finding "good" solutions to large-scale (possibly otherwise intractable) discrete

optimization problems. They do not require a closed form of the objective function, relying only on value of the function at various points which can be estimated via simulation. While they are not guaranteed to produce optimal solutions, they often produce good solutions in a reasonable amount of time and are included in some commercial simulation software packages. Examples of meta-heuristics include simulated annealing, genetic algorithms, tabu search, scatter search, and the learning automata search technique [61, 134]. A brief description of each method follows.

Simulated annealing is an adaptive search method that "exploits the analogy between the way in which metal cools and then freezes into a minimum energy crystalline structure and the search for a minimum in a more general system" [147]. In simulated annealing, search directions (or points, in the discrete case) are randomly generated. If a point with a lower function value is found, then it becomes the new iterate. Otherwise, it may still randomly accept the point with nonzero probability, based on the function values and a control parameter. While it may seem odd to accept a worse point, doing so can help avoid local (non-global) minima [128,134,147].

Genetic algorithms, one class of evolutionary algorithms, mirror the evolutionary processes found in nature. In nature, each species searches for beneficial adaptation in a changing environment. As species evolve, new attributes are encoded in the chromosomes of its members. Though this information can change by mutation, most change occurs due to the combination and exchange of genetic attributes during breeding. Analogously, genetic algorithms code complicated non-biological structures as simple bit strings and improve the structures by simple transformations of the strings. They are unique in that they work on encoded decision variables rather than the variables themselves, and search from one population to another rather than from one individual to another [147]. Thus, each new population has a good chance of inheriting the best characteristics of the previous one [134]. Genetic algorithms are generally effective, robust, computationally simple [128], and easy to parallelize [134, 147]. However, they are often very expensive, in that they generally require many function evaluations.

Tabu search, proposed by Glover in 1977, is described as "a meta-heuristic superimposed on another heuristic" [57]. The crux of tabu search is to avoid getting caught in cycles or local optima by forbidding or penalizing moves which steer the solution towards regions of solution space already visited. As with other meta-heuristics, it is partially motivated by an observation of nature, human nature in this case. Human behavior appears to operate with a random element that leads to inconsistent behavior in similar situations. It is this inconsistent behavior or deviation from a charted course that proves to be a source of gain. Tabu search works in this way except that it does not randomly choose new courses. Instead it only accepts a "poor" course if it is the only way to avoid a path previously studied. In this way it can potentially escape the trap of local minima [128, 131].

Scatter search is a population-based method originally proposed by Glover in the 1960s. Unlike other evolutionary algorithms, it does not search in a completely random fashion. It instead uses strategies for combining effective solution vectors through the use of an adaptive memory similar to that used by tabu search [18, 19, 58, 103].

The learning automata search technique (LAST) is based on game theory and artificial intelligence. Unlike many other meta-heuristics, LAST is not a local search technique. Instead, it jumps around the solution space while keeping the best point. Initially, it jumps randomly, but as the algorithm progresses, it begins to "zero in" on its perceived best solution by updating the probabilities associated with the random jump. Initially, the algorithm has an equal probability of choosing various points to evaluate. At each iteration, a point is selected based on probabilities that are stored with each previously evaluated point, and evaluated. If the selected point has a comparatively low objective function value, then the corresponding probability is rewarded with an increase. Otherwise, it is punished with a decrease. In this way the algorithm "gets smarter" [61, 164]. LAST has also been combined with a genetic algorithm to speed solution convergence properties and increase its chance of escaping local optima [70].

2.1.2.4 Mixed Variable Solution Spaces. Often in real-world problems the solution space is not neatly defined as either discrete or continuous, but rather a combination of both types, *i.e.*, it is a problem in which some decision variables are discrete and some are continuous. Historically, methods to tackle this problem have included 1) discretize the solution space of the continuous functions using a reasonably fine grid, 2) treat the discrete variables as continuous and use continuous optimization algorithms with rounding, or 3) use branch-and-bound techniques branching only on the discrete variables. Naturally, these methods have drawbacks. With the first option, the design space can become quite large depending on the size of the grid, while with the second, treating discrete variables as continuous is known to have problems in practice, often leading to sub-optimal or infeasible solutions. Finally, branch-and-bound techniques require relaxation of integrality constraints [162] and so are not applicable to problems involving categorical variables. Solution methods for this type of problem include pattern search methods for solving deterministic mixed variable problems introduced by Audet and Dennis [9]. Recent work by Sriver *et al.* [138] extended the work of Audet and Dennis to the stochastic case using a combination of pattern search and ranking and selection. This method (MGPS-RS) is discussed further in Chapter III. A more general framework for treating mixed variable problems (without the specifying the approach for handling the continuous variables) was introduced by Lucidi *et al.* for both the derivative-free case [95] and the case in which derivatives are employed [94].

2.2 Methods for Multi-objective Optimization

There are many methods available to solve multi-objective optimization problems. These methods can be sorted into three families: *a priori* methods, progressive methods, and *a posteriori* methods. With *a priori* methods, the decision maker defines the trade-off to be applied (preferences) before running the optimization method. With progressive methods, the decision maker improves the trade-off as the method progresses. And finally, in *a posteriori* methods, the decision maker chooses a solution

after examining all those obtained by the optimization method. Not all methods fall neatly into one family. For example, *a priori* preferences can be computed at random and then a large number of solutions can be presented to the decision maker to choose the final solution.

Most multi-objective optimization methods can also be divided into five sets: scalar methods, interactive methods, fuzzy methods, meta-heuristic methods, and decision aid methods, as shown in Figure 2.1(a). Again, not all methods can be pigeon-holed into a single set. For example, the method of interactive specification of aspiration/reservation levels, discussed extensively in Chapter III, is a scalar method that is also interactive. Though not absolute, the sets do provide a useful way of categorizing and describing most methods. A description and examples of each set of methods follows. Additional information on multi-objective solution methods can be found in [33].

2.2.1 Scalar Methods. Scalar methods attempt to transform the multi-objective problem into one with a single objective, so that standard optimization techniques can be applied. Several methods are presented.

2.2.1.1 Weighted Sum of the Objective Functions. This approach is the most obvious of the solution methods and is sometimes called the "naïve approach". It simply constructs the single objective as a weighted sum of the objective functions. The problem can be solved repeatedly using different different weights to generate an approximation of the Pareto front. This was the first solution method used on multi-objective problems; though it is quite efficient, it may perform poorly on problems with nonconvex Pareto fronts. If the Pareto front is nonconvex, not all solutions on the tradeoff surface can be found [33,37,128]. For example, the Pareto front depicted in Figure 1.2 is an extreme case where only the two endpoints of the front can be found with this type of method, regardless of the weights [32].

2.2.1.2 Keeney-Raiffa Method. The Keeney-Raiffa method forms a single objective function as the product of the objective functions. The new objective function is called the Keeney-Raiffa utility function and is used in multi-attribute utility theory, which comes from the decision sciences and deals with the properties of a utility function and ways to create it [33].

2.2.1.3 Distance-to-a-Reference Point Method. This method also transforms the objective functions into one by calculating the distance to a reference point, such as the utopia point, using an appropriate distance metric (*e.g.*, the Euclidean norm). A Pareto optimal point is then found by minimizing the distance to the reference point. The success of this method is heavily dependent on the choice of a reference point, but it is often able to find solutions hidden in concavities of the Pareto front [33].

2.2.1.4 Compromise Method. While the previous three methods merge several objective functions into a single one, the compromise method does so with additional constraints. This is done by preserving the highest priority objective function (as specified by the decision maker) and transforming all others into constraints. Though the compromise method is known to consume large amounts of computing time and the programming of the algorithm can be difficult with many objective functions, the relative simplicity of its equations has made it popular [33].

2.2.1.5 Normal Boundary Intersection Method. The normal boundary intersction (NBI) approach of Das and Dennis [38] produces an approximation of the Pareto front by solving a series of single-objective optimization problems, in which an additional linear equality constraint based on previously determined objective function values is added [12]. This results in a Pareto solution that lies on the intersection of the constraint and the Pareto front. It is based on the observation that the intersection of the boundary of the set of feasible solutions (or image of the feasible region) with a normal pointing towards the origin and emanating from any

point in the convex hull of individual minima (convex combination of the objectives) is a point on the portion of the boundary which contains efficient points. NBI has been shown to provide an evenly distributed set of points on the Pareto front [37].

2.2.1.6 Hybrid Methods. Hybrid methods combine several methods into new ones. The best known hybrid method is the Corley Method, which converts some objectives to constraints and then applies a weighted-sum-of-the-objective-functions to the remaining objective functions. This method is efficient in solving various kinds of optimization problems, with both convex and nonconvex Pareto fronts, but has twice the number of parameters to "tune"(*e.g.*, weights, which objectives to convert to constraints, and in what order) [33].

2.2.1.7 Goal Attainment Method. The goal attainment method is relatively simple and is able to determine points on nonconvex regions of the Pareto front (see Figure 1.1). The methods consists of choosing an initial vector of decision-maker-specified ideal function values, choosing a search direction (essentially weights or relative importance of the objectives), and minimizing a scalar coefficient which represents the gap between the two vectors. The main advantage of this method is its efficiency with respect to nonconvex Pareto fronts; however, some shapes of Pareto fronts exist for which the success of the method depends greatly on the choice of the search direction [33]. An example is the method of interactive specification of aspiration/reservation levels, mentioned in Section 2.2.2.6, which is an interactive/-goal attainment method that uses aspiration levels (decision-maker-specified ideal or "hoped for") as the reference point. The search direction is determined by the ray between the aspiration level and reservation level (decision-maker-specified "worst case scenario") [96]. This method is discussed further in Section 3.1.4.

2.2.1.8 Goal Programming Method. This method is similar to the goal attainment method, except that the transformed problem has equality instead of inequality constraints. Because it is so similar, this method has the same relative

advantages and disadvantages as the goal attainment method [33].

2.2.1.9 Lexicographic Method. As the name implies, this method takes the objectives and orders them by preference. The mono-objective problems are then solved one at a time while converting already solved objectives into constraints. Though intuitive and easily solved, the main drawback of this method is that an appropriate sequence of the objective functions must be determined without inadvertently excluding promising regions of the space prematurely [33, 128].

2.2.2 Interactive Methods. Interactive methods belong to the progressive methods family and thus allow the decision maker to tune preferences with regard to tradeoffs as the methods progress. These methods find one and only one solution. Several interactive methods are presented below.

2.2.2.1 Surrogate-Worth Tradeoff Method. This method adds an interactive process to the compromise method (see Section 2.2.1.4) to progress towards a solution. Since the decision maker has input throughout the process, a solution found via this method has a good chance of being satisfactory. However, the choice of opinion values can greatly affect the solution quality, and because it relies on gradient information, this method should not be applied to problems for which the objective functions are not differentiable [33, 106].

2.2.2.2 Fandel Method. The goal of the Fandel method is to assist the decision maker in choosing the weights used in the weighted-sum-of-the-objective-functions method. Each objective function is first optimized separately to generate the utopia point. This point and the feasible region are then used to calculate a domain of variation (region of the objective space bounded by the minimum and maximum values of each objective). An attempt is then made to find the hyperplane parallel to the one passing through the extremities of the domain of variation and tangent to the feasible space. The tangent point provides a vector that corresponds

to weights that are used to determine a solution. The decision maker either accepts this solution or the search space is reduced. The main drawback of the Fandel method is that parts of the tradeoff surface cannot be approximated, and it only finds a single solution. Additionally, it assumes that the solution that satisfies the decision maker is the one closest to the ideal point, which may not be the case [33].

2.2.2.3 Step Method. This method is similar to the Fandel method in that information about the preferences of the decision maker allows the search space to be restricted at each iteration. The decision maker indicates an upper bound for acceptable values of the objective functions and then weights are calculated such that objectives with a larger range of possible values are given larger weights in the weighted-sum-of-the-objective-functions. The problem is optimized and the new solution is evaluated by the decision maker. If it is not acceptable, the upper bound of the solution space can be adjusted and the weights recalculated. Compared with the Fandel method, the weights have less importance because interaction with the decision maker is done via selection of the desired upper bound and not the weights directly [33].

2.2.2.4 Jahn Method. In contrast to the Fandel and STEP methods, the Jahn method depends on both the decision maker preferences and gradient information. It begins by choosing a starting point and then stepping through the solution space using a gradient-based method and decision maker preferences toward a Pareto optimal point. Since gradient information is required, the objective functions must be differentiable [33].

2.2.2.5 Geoffrion Method. This method is similar to the Jahn method and is based on the Frank-Wolfe algorithm. The method begins with a weighted-sums-of-the-objective-function step. The resulting point is then used to linearize the objective function and compute a direction that reduces the objective while maintaining feasibility [52]. In the next step instead of performing a line search to find

the optimal step along that direction (as in the Frank-Wolfe method), the step size is determined by the decision maker. The new solution is either accepted by the decision maker or the process is repeated [33].

2.2.2.6 Interactive Specification of Aspiration/Reservation Levels and Achievement Scalarization Functions. As mentioned in Sections 2.2 and 2.2.1.7, this method is based on idea that a decision maker has a desired level for each objective, as well as a level beyond which the solution is not acceptable. These values, called *aspiration levels* and *reservation levels*, respectively, are used in a method similar to distance-to-a-point or compromise methods (where the aspiration level is the reference point and both are used to determine the search direction) in order to find a single solution. The decision maker can then either accept the solution or revise the aspiration and reservation levels [96]. This method is discussed further in Chapter III.

2.2.3 Fuzzy Methods. Though real world phenomenon are rarely binary in nature, for a long time the only tool for describing them was binary logic, written in terms of *true* or *false*. In response, L.A. Zadeh developed a new logic based on *fuzzy sets*, which not only offers a way to model uncertainty and imprecision, but also accounts for progressive transition between states, *e.g.* shades of gray between black and white. This is accomplished with a *membership function*, which relates each decision variable to a continuous variable varying between zero and one and indicates the degree to which the variable belongs to the set represented by the original logical decision variable and allowing a progressive transition between *true* and *false*. Two fuzzy logic methods are briefly presented [33, 128].

2.2.3.1 Sakawa Method. This method uses fuzzy logic at all levels: the parameters of the problem, constraints, and solution set. Typical of fuzzy logic methods, the solutions have a membership function that is correlated with the original objective function at a level set by the decision maker. This method allows

the decision maker to specify the number of solutions desired, which is useful when the decision maker is more demanding [33]. Sakawa also combined fuzzy logic with evolutionary methods in order to handle multi-objective binary programming problems [129]. Despite its interesting advantages, the Sakawa method is cumbersome and uses fuzzy sets and mathematical rules that are difficult to apply [33].

2.2.3.2 Reardon Method. In contrast, the Reardon method is a relatively simplified fuzzy logic method. Its membership function is simply shaped and easily obtained compared to the Sakawa method. It has been shown to give good results on two test problems, Schaffer F2 test problem and the simplified Born-Mayer test problem [33] and that, in contrast to genetic algorithms like the niched Pareto approach (see Section 2.2.4.4), the efficiency of the algorithm is independent of the number of objectives [120, 121].

2.2.4 Multi-objective Methods Using Meta-Heuristics. Meta-heuristics are general optimization methods which need some transformation before being applied to the solution of a particular problem and are often analogous to real life concepts [128]. They are usually applied to "hard" optimization problems. As discussed in Section 2.1.2.3, meta-heuristics are not guaranteed to produce optimal solutions, but they often produce good solutions in a reasonable amount of time [33]. Four meta-heuristics are presented: simulated annealing, tabu search, scatter search and genetic (evolutionary) algorithms.

2.2.4.1 Simulated Annealing. Simulated annealing, discussed previously for stochastic optimization in Section 2.1.2.3, has also been used in multi-objective optimization [141]. Examples of multi-objective simulated annealing methods include Multiple Objective Simulated Annealing (MOSA) [33,47,93], Fuzzy Pareto Simulated Annealing (FPSA) [133], and Pareto Archived Simulated Annealing (PASA) [33]. MOSA uses simulated annealing to search the tradeoff surface, considering objectives separately to determine the probability of accepting a bad solution but using

a weighted-sum-of-the-objective-functions approach to calculate the actual solution fitness (*i.e.* the value of the weighted objective function). The method behaves well because at a high temperature, the simulated annealing spreads the solution population over the whole tradeoff surface [33]. If the temperature is "sufficiently high" at the end of the simulation, MOSA can find solutions unreachable by the weighted-sum-of-the-objective-functions method [147]; however, what constitutes "sufficiently high" is completely problem-dependent is not generally known *a priori*. In FPSA, simulated annealing is combined with fuzzy logic (see Section 2.2.3) in order to find an approximation of the Pareto set [65]. PASA uses an aggregated function of objective functions, coupled with a system that stores nondominated solutions separately from the rest of the iterates and uses them to assess solution quality [33].

2.2.4.2 Tabu Search. Tabu search, discussed previously for stochastic optimization (Section 2.1.2.3), has been successfully used in multi-objective searches for continuous problems [33, 55]. Implementations include M-Objective Tabu Search [28], MOTS [47, 64], and New Multi-objective Tabu Search (NMTS) [118]. Tabu search has also been used in conjunction with scatter search algorithms, *e.g.*, Scatter Tabu Search Procedure for Multiobjective Optimization (SSPMO) [103].

2.2.4.3 Scatter Search. Scatter search, discussed previously for stochastic optimization (Section 2.1.2.3), has been successfully used in multi-objective searches [18, 58, 97, 98]. Implementations include the Scatter Tabu Search Procedure for Multiobjective Optimization (SSPMO) [103] and Multi-Objective Scatter Search (MOSS) [19].

2.2.4.4 Genetic/Evolutionary Algorithms. Genetic algorithms, a type of evolutionary algorithm, were also discussed as a stochastic optimization method in Section 2.1.2.3. They are useful for multi-objective problems because they deal simultaneously with a set of possible solutions (the population) which allows the algorithm to find several elements of the Pareto optimal set in a single run. Unlike many tra-

ditional math programming techniques (*e.g.*, aggregation methods), they can handle discontinuous and concave Pareto fronts. A multi-objective genetic algorithm differs from single-objective ones in how they evaluate solutions. Unlike single-objective genetic algorithms, multi-objective ones cannot directly use the objective function value as a measure of solution fitness, so some combination of the objectives must be used. Many methods have been proposed, and multi-objective evolutionary algorithms can be classified by how they determine if a solution is efficient: aggregating functions, population-based approaches, and Pareto-based approaches [32, 134].

Aggregating Functions. As discussed in Section 2.2.1, aggregating functions are probably the most straightforward of approaches, but certain problems exist. Linear aggregating functions, a weighted sum for example, cannot generate nonconvex portions of the Pareto front no matter what weights are chosen. However, nonlinear functions—*e.g.*, achievement scalarization functions (see Section 3.1.4 and [96]), the Keeney-Raiffa method (see Section 2.2.1.2), and the goal attainment method (see Section 2.2.1.7)—do not suffer from this limitation and have been used successfully in multi-objective combinatorial optimization [32].

Population-Based Approaches. Population-based approaches use the population of the evolutionary algorithm to diversify the search and potentially find multiple Pareto optimal points at each iteration, but the idea of dominance is not directly incorporated into the selection process. An example of this type of approach is the Vector Evaluated Genetic Algorithm (VEGA) developed by Schaffer [132]. VEGA consists of a simple genetic algorithm with a modified selection mechanism. At each generation, k sub-populations of size N/k, where N is the total population size, are generated based on proportional selection according to each objective function, in turn [132]. These sub-populations are then shuffled together to form one population to which crossover and mutation operators are applied. VEGA has several problems, the most serious of which is that the selection method it uses does not accurately incorporate Pareto dominance. If, for example, a good compromise

(Pareto optimal) solution is generated, but it is not the best solution for any given objective function, it will be discarded. Some methods have been developed to help alleviate this, but the underlying problem remains [32, 47]. Other population-based approaches include the Random Sampling Evolutionary Algorithm (RAND) [165] and Hajela and Lin's genetic algorithm (HLGA) [63, 165].

Pareto-Based Approaches. In contrast to population-based approaches, this type of approach incorporates the concept of Pareto optimality directly into the selection method of the evolutionary algorithm by using dominance rather than an aggregated objective function to measure fitness. Thus, the Pareto-based approaches do not suffer the problems associated with VEGA. Many Pareto-based approaches have been developed [32], including Goldberg's Pareto Ranking [59], Multi-Objective Genetic Algorithm (MOGA) [47, 51], the Nondominated Sorting Genetic Algorithm (NGSA and NGSA-II) [42, 136, 165], Niched Pareto Genetic Algorithm (NPGA) [68, 165], Strength Pareto Evolutionary Algorithm (SPEA and SPEA2) [47, 165], Multi-Objective Messy Genetic Algorithm (MOMGA, MOMGA-II, and MOMGA-IIa) [40], Multi-Objective Hierarchical Bayesian Optimization Algorithm (hBOA) [32], Pareto Archived Evolutionary Strategy (PAES) [47, 79, 112], Micro-Genetic Algorithm for Multi-Objective Optimization [31], and the General Multi-Objective Program (GENMOP) [78, 125].

2.2.5 Decision Aid Methods. Decision aid methods differ from the others in several ways. Instead of filtering the elements of the solution set and keeping those elements that could be compared to themselves (*e.g.*, lexicographic optimality), an order relation is set up among elements of the solution set and thus a set of solutions (with a partial order relation, *i.e.*, a relation that is reflective, asymmetric, and transitive [153]), or a single solution (with a total order relation, *i.e.*, a relation that is reflective, asymmetric, transitive, and total[1] [154, 160]), is obtained. Also,

[1]A binary relation R over a set X is *total* if it holds for all a and b in X that a is related to b or b is related to a (or both), *i.e.*, $\forall a, b \in X$, $aRb \vee bRa$ [161].

these methods only work on discrete sets of points. The decision aid methods choose or sort actions with respect to a set of criteria. Each method produces its own definition of the value of a criterion [33]. Decision aid methods include ELECTRE methods (I, IS, II, III, IV, and TRI) [43, 56, 130] and PROMETHEE (I and II) [130].

2.3 Methods for Stochastic Multi-objective Optimization

Generally, there are two types of stochastic optimization problems: those with random noise in the function evaluations, and those with random algorithm choices (such as the search direction) [134]. Most stochastic multi-objective problems solved to date are of the second type. Examples include methods that use evolutionary algorithms (see Section 2.2.4.4). Relatively few multi-objective solution methods of the first type have been developed. Three methods proposed by Baba and Morimoto [13] are discussed in Sections 2.3.1, 2.3.2, 2.3.3. A more recent method developed by Audet *et al.* [12] is discussed in Section 2.3.4. This method has not yet been applied to the stochastic case.

2.3.1 Learning Automata. Learning automata is a reinforcement (or feedback) learning scheme where actions by the automaton produce results for which either reward or punishment ensue. The feedback then changes the probability of choosing that action such that rewards increase the probability of selecting that action again, and punishment decreases that probability. This process is repeated until the automaton "learns" what actions are optimal [61, 134]. In the multi-objective case, Baba and Morimoto show that an appropriately chosen learning scheme ensures convergence to a "reasonable solution" for a finite number of candidate solutions [13].

2.3.2 Random Optimization Method. Random search methods—typically used for single objective problems—are based on searching the problem domain in a random manner in order to optimize an objective. These methods are the simplest of stochastic optimization, but can be effective in certain cases [134]. Baba and Morimoto showed that their random optimization algorithm is guaranteed to converge

almost surely to a Pareto-optimal solution but only under strict assumptions on the decision space, solution space, and error. They suggest further study to find a less restrictive approach [13].

2.3.3 Stochastic Approximation Method. Baba and Morimoto [13, 14] propose a stochastic quasigradient method to solve the stochastic multi-objective optimization problem. Subgradients are used for convex functions that are not necessarily differentiable at all points. For the convex function $f : \Omega \rightarrow \mathbb{R}$, the set of subgradients at x_n in the open interval Ω is $G_n = \{c \in \mathbb{R} : f(x) - f(x_n) \geq c(x - x_n), \forall x \in \Omega\}$ [159]. A stochastic quasigradient is a stochastic estimate of the subgradient of the function at the point x_n, *i.e.*, given a set of points $\{x_0 \dots x_n\}$, the stochastic subgradient ξ_n is a random vector such that $E(\xi_n | x_0 \dots x_n) = a_n \hat{F}_x(x_n) + b_n$ where the vector $\hat{F}_x(x_n)$ is the subgradient of the function $F(x)$ at the point $x = x_n$, b_n is a $\mathcal{H}$-valued random variable, $\mathcal{H}$ is a separable Hilbert space, and a_n is a positive real-valued random variable. [48, 49, 116]. Under assumptions of continuity, compactness, and bounded error, Baba and Morimoto show that the algorithm extended to the multi-objective case converges with probability one to the global solution [14].

2.3.4 Bi-Objective Mesh Adaptive Direct Search (BiMADS). Audet *et al.* have extended the Mesh Adaptive Direct Search (MADS) (discussed in Section 3.1.3) framework to apply to the bi-objective case. Their algorithm, BiMADS, is similar to reference point methods (see Section 2.2.1.3), but also takes advantage of the ordering property of the Pareto front inherent only in bi-objective problems [12, 22]. BiMADS is shown to produce solutions satisfying some first order necessary conditions for optimality based on the Clarke calculus [30], and like the NBI method (see Section 2.2.1.5), attempts to generate evenly distributed points on the Pareto front [12]. Thus far BiMADS has been developed for deterministic problems; however, the extensions of MADS to the mixed variable and stochastic case discussed in Chapter III would also apply to BiMADS.

2.4 More Generalized Stochastic Multi-objective Optimization Method

As seen throughout this chapter, many solution methods exist for both stochastic optimization and multi-objective optimization when only one of the two characteristics exists in the problem. As discussed in Section 2.3, four methods exist for problems having both characteristics and because of necessary assumptions on the decision and objective spaces, such methods are applicable only to a small number of problems. None of these methods is applicable to stochastic, multi-objective, mixed-variable optimization problems as considered in this research. Thus, a more generally applicable method is required and is presented in Chapter III.

III. Research Methodology

3.1 Components for a New Algorithm

In this chapter, an algorithm is presented for solving stochastic mixed variable optimization problems with multiples objectives. This algorithm relies on a number of important features that are described here in greater detail. Particularly, consider the following observations. Generalized pattern search with ranking and selection (MGPS-RS) has been successfully developed for stochastic, linearly constrained mixed-variable problems [137, 138] and has been applied to a multi-echelon repair system [144]. Mesh adaptive direct search (MADS) has been developed for deterministic, continuous, nonlinearly constrained problems [11]. However, MGPS-RS and MADS in their current forms apply only to single-objective problems. Alternatively, interactive techniques using aspiration/reservation levels and scalarization functions have been used successfully to find Pareto optimal solutions to deterministic multi-objective problems [62]. Finally, a multi-objective ranking and selection technique by Lee *et al.* [85], called *multi-objective optimal computing budget allocation* (MOCBA), has been applied to selecting the non-dominated set of inventory policies for aircraft maintenance, a discrete variable problem [86].

The rest of this section gives a more detailed description of each of the methods mentioned thus far. A new algorithm, called *Stochastic Multi-Objective Mesh Adaptive Direct Search* (SMOMADS), is then proposed in Section 3.2, which combines elements of these methods to handle the class of mixed variable, stochastic, multi-objective optimization problems. A convergence analysis of the new algorithm then follows in Section 3.3.

3.1.1 Ranking and Selection. Problems having stochastic responses can pose a particular challenge for optimization algorithms that rely solely on comparisons of trial points. Noise in the response can lead to errors in iterate selection if the observed responses are comparatively different than the true function values. Methods for selecting a "best" among several trial points must account for variation to provide

some statistical assurance of correct selection. Achieving high confidence in the correctness of selection requires additional replications (*i.e.*, function evaluations) at each trial point. Ranking and selection (R&S) considers multiple candidates simultaneously at a reasonable cost. This is possible because R&S detects a relative order of the candidates instead of generating precise estimates [139]. The discussion of R&S that follows, including the notation, mirrors that of [139].

Let X_k denote the k-th element of a sequence of random vectors and x_k denote a realization of X_k. For a finite set of candidate points $C = \{Y_1, Y_2, \ldots, Y_{n_C}\}$ with $n_C \geq 2$, let $f_q = f(Y_q) = E[F(Y_q, \cdot)]$ denote the *true* mean of the response function F at Y_q for each $q = 1, 2, \ldots, n_C$. These response means can be ordered (from minimum to maximum) as

$$f_{[1]}, f_{[2]}, \ldots, f_{[n_C]}. \tag{3.1}$$

Denote by $Y_{[q]} \in C$ the candidate from C with the q-th lowest true objective function value.

Given some $\delta > 0$, called the *indifference zone parameter*, no distinction is made between two candidate points whose true means satisfy $f_{[2]} - f_{[1]} < \delta$. In such cases, the method is said to be *indifferent* in choosing either candidate as the best. The probability of correct selection (CS) is defined as

$$P\{CS\} = P\left\{\text{select } Y_{[1]} | f_{[q]} - f_{[1]} \geq \delta;\ q = 1, 2, \ldots, n_C\right\} \geq 1 - \alpha, \tag{3.2}$$

where $\alpha \in (0, 1)$ is the significance level. Because random sampling guarantees that $P\{CS\} = \dfrac{1}{n_C}$, the significance level must satisfy $0 < \alpha < 1 - \dfrac{1}{n_C}$.

Because the true objective function values are not available, it is necessary to work with the sample means of F. For each $q = 1, 2, \ldots, n_C$, let s_q be the total number of replications at Y_q and let $\{F_{qs}\}_{s=1}^{s_q} = \{F(Y_{qs}, W_{qs})\}_{s=1}^{s_q}$ be the set of simulated responses, where $\{Y_{qs}\}_{s=1}^{s_q}$ are the replications at candidate point Y_q, and W_{qs} are realizations of the random noise. For each $q = 1, 2, \ldots, n_C$, the sample mean

Procedure $RS(C, \alpha, \delta)$

Inputs: $C = \{Y_1, Y_2, \ldots, Y_{n_C}\}$, $\alpha \in (0,1)$, $\delta > 0$.

Step 1: For each $Y_q \in C$, use an appropriate statistical technique to determine the number of samples s_q required to meet the probability of correct selection guarantee in Equation (3.2), as a function of α, δ and respoinse variation of Y_q.

Step 2: For each $q = 1, 2, \ldots, n_C$, obtain replicated responses F_{qs}, $s = 1, 2, \ldots, s_q$, and compute the sample mean $\overline{F}_q$, according to Equation (3.3).

Return: $\hat{Y}_{[1]} = \arg\left(\overline{F}_{[1]}\right)$.

Figure 3.1: A generic R&S Procedure (see Figure 1 in [139])

$\overline{F}_q$ is given by

$$\overline{F}_q = \frac{1}{s_q} \sum_{s=1}^{s_q} F_{qs}. \tag{3.3}$$

The sample means can be ordered and indexed in the same manner as in Equation (3.1), and let $\hat{Y}_{[q]} \in C$ denote the candidate with the q-th lowest *estimated* objective function value as determined by the R&S procedure. The candidate corresponding to the minimum mean response, $\hat{Y}_{[1]} = \arg\left(\overline{F}_{[1]}\right)$, is chosen as the best point. A generic R&S procedure is shown in Figure 3.1. The algorithmic choices made in Step 1 define different R&S methods. See [110, 115, 122], for examples.

3.1.2 MGPS-RS. Pattern search algorithms are defined through a finite set of directions used at each iteration. The direction set and a step length parameter are used to construct a discrete set of points, or mesh, around the current iterate. The mesh at iteration k is defined to be

$$M_k = \bigcup_{x \in S_k} \{x + \Delta_k^m D z : z \in \mathbb{N}^{n_D}\}, \tag{3.4}$$

where S_k is the set of points where the objective function f has been evaluated by the start of iteration k (S_p is the set of initial points), Δ_k^m is called the *mesh size parameter*, and D is a positive spanning set (see [39]); *i.e.*, a set of directions that

positively spans $\mathbb{R}^n$. Additional restrictions on D are that each direction $d \in D$, $j = 1, 2, \ldots, n_D$, must be the product of some fixed nonsingular generating matrix $G \in \mathbb{R}^{n \times n}$ and an integer vector $z_j \in \mathbb{Z}^n$ [146]. For bound and linearly constrained problems, the directions in must D be sufficiently rich to ensure that polling directions can be chosen that conform to the geometry of the constraint boundaries, and that these directions be used infinitely many times [2]. A finite set of trial points called the *poll set* is then chosen from the mesh, evaluated, and compared to the incumbent solution. If improvement is found, the incumbent is replaced and the mesh is retained or coarsened via the mesh size parameter Δ_k^m. If not, the mesh is refined and a new set of trial points is selected.

Initially developed by Torczon [146], GPS was extended by Lewis and Torczon to include bound [87] and linear [88] constraints. GPS was further extended by Lewis and Torczon [89] to include nonlinear constraints using an augmented Lagrangian approach by Conn, Gould, and Toint [34], and by Audet and Dennis [10], via a filter [50]. Dissatisfaction with the convergence theory of filter-GPS led to the creation of MADS [11] (see section 3.1.3). GPS for bound constrained mixed variable problems (or mixed variable pattern search – MVPS) was introduced separately by Audet and Dennis [9], and extended to linearly constrained by problems by Abramson [2]. It was then extended further to general nonlinearly constrained MVP problems, also via a filter, by Abramson, Audet and Dennis [2, 6].

The GPS framework, in conjunction with ranking and selection, was used by Sriver [137, 139] to handle stochastic MVP problems. In this case, the poll set at each iteration is given by $P_k(x_k) \bigcup \mathcal{N}(x_k)$, where $\mathcal{N}(x_k)$ is a user-defined set of discrete neighbors around x_k and

$$P_k(x) = \left\{ x + \Delta_k^m (d, 0) : d \in D_k^i \right\}, \tag{3.5}$$

where $(d, 0)$ denotes that the variables have been partitioned into continuous and discrete variables, with the discrete variables remaining unchanged. The set of discrete

neighbors is defined by a set-valued function $\mathcal{N} : \Omega \to 2^{\Omega}$, where 2^{Ω} denotes the power set of Ω. The notation $y \in \mathcal{N}(x)$ means that the point y is a discrete neighbor of x. By convention, $x \in \mathcal{N}(x)$ for each $x \in \Omega$, and it is assumed that $\mathcal{N}(x)$ is finite [137]. A generic indifference-zone ranking and selection procedure $RS(P_k(x_k) \bigcup \mathcal{N}(x_k), \alpha, \delta)$, with indifference-zone parameter δ and significance level α, is used to select among points in the poll set for improved solutions, *i.e.*, δ-near-best mean. The rules for updating the mesh size are as follows [11]. Given a fixed rational number $\tau > 1$ and two integers $m^- \leq -1$ and $m^+ \geq 0$, the mesh size parameter Δ_k^m is updated according to the rule,

$$\Delta_{k+1}^m = \tau^{w_k} \Delta_k^m, \tag{3.6}$$

where

$$w_k \in \begin{cases} \{0, 1, \ldots, m^+\}, & \text{if an improved mesh point is found} \\ \{m^-, m^- + 1, \ldots, -1\}, & \text{otherwise.} \end{cases} \tag{3.7}$$

If no improvement can be found, an extended poll step is conducted to search amongst any discrete neighbor $y \in \mathcal{N}(x_k)$ that satisfies $f(y) < f(x_k) + \xi_k$, where ξ_k is called the *extended polled trigger*. Each neighbor meeting this criteria, in turn, becomes the poll center, and the extended poll continues until either a better point than the best iterate is found, or else they are all worse than the extended poll center [9, 137]. Sriver showed that this algorithm generates an iteration subsequence with almost sure convergence to a stationary point "appropriately defined" in the mixed-variable domain [138]. The MGPS-RS Algorithm is shown in Figure 3.2.

3.1.3 Mesh Adaptive Direct Search. Mesh Adaptive Direct Search (MADS) is a class of algorithms developed by Audet and Dennis for minimization of nonsmooth functions of the type $f : \mathbb{R}^n \to \mathbb{R} \bigcup \{+\infty\}$ under general constraints $x \in \Omega \subseteq \mathbb{R}^n$ and $\Omega \neq \emptyset$. The feasible region Ω may be defined by blackbox constraints, *e.g.*, computer code that returns a yes/no answer to whether or not a trial point is feasible [11].

A General MGPS-RS Algorithm

- INITIALIZATION: Let $X_0 \in \Omega$, $\Delta_0^m > 0$, $\xi > 0$, $\alpha_0 \in (0,1)$ and $\delta_0 > 0$. Set the iteration and R&S counters $k = 0$ and $r = 0$ respectively.
- POLL STEP: Set extended poll trigger $\xi_k \geq \xi$. Use R&S procedure $RS(P_k(X_k) \bigcup \mathcal{N}(X_k), \alpha_r, \delta_r)$ to return the estimated best solution $\hat{Y}$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$. If $\hat{Y} \neq X_k$, the step is *successful*, update $X_{k+1} = \hat{Y}$, $\Delta_{k+1}^m \geq \Delta_k^m$ according to Equations (3.6) and (3.7), and $k = k + 1$ and return to POLL STEP. Otherwise, proceed to EXTENDED POLL STEP.
- EXTENDED POLL STEP: For each discrete neighbor $Y \in \mathcal{N}(X_k)$ that satisfies the extended poll trigger condition $\overline{F}(Y) < \overline{F}(X_k) + \xi_k$, set $j = 1$ and $Y_k^j = Y$ and do the following.
  - Use R&S procedure $RS(P_k(Y_k^j), \alpha_r, \delta_r)$ to return the estimated best solution $\hat{Y}$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$. If $\hat{Y} \neq Y_k^j$, set $Y_k^{j+1} = \hat{Y}$ and $j = j + 1$ and repeat this step. Otherwise, set $Z_k = Y_k^j$ and go to the next step.
  - Use R&S procedure $RS(X_k \bigcup Z_k), \alpha_r, \delta_r)$ to return the estimated best solution $\hat{Y}$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$. If $\hat{Y} = Z_k$, the step is successful, update $X_{k+1} = \hat{Y}$, $\Delta_{k+1}^m \geq \Delta_k^m$ according to Equations (3.6) and (3.7), and $k = k + 1$ and return to the POLL STEP. Otherwise, repeat the EXTENDED POLL STEP for another discrete neighbor that satisfies the extended poll trigger condition. If no such discrete neighbors remain in $\mathcal{N}(X_k)$, set $X_{k+1} = X_k$, $\Delta_{k+1}^m < \Delta_k^m$ according to Equations (3.6) and (3.7), and $k = k + 1$ and return to the POLL STEP.

Figure 3.2: The Mixed-variable GPS Ranking and Selection (MGPS-RS) Algorithm [138]

MADS is similar to MGPS-RS in the generation of the mesh and poll sets (see Equations (3.4) and (3.5) in Section 3.1.2) as well as in the rules for updating the mesh (see Equations (3.6) and (3.7)). However, the key difference is that in MADS [11] a *poll size parameter* Δ_k^p is introduced, which satisfies $\Delta_k^m \leq \Delta_k^p$ for all k such that

$$\lim_{k \in K} \Delta_k^m = 0 \quad \Leftrightarrow \quad \lim_{k \in K} \Delta_k^p = 0 \text{ for any infinite subset of indices in } K. \tag{3.8}$$

The poll size parameter controls the magnitude of the distance between the incumbent solution, and the set of directions used to define the poll set are constructed in a different manner. In GPS, only one value $\Delta_k = \Delta_k^p = \Delta_k^m$ is used, and a set of positive spanning directions $D_k \subset D$ is chosen at each iteration. In the poll step of MADS, both of these restrictions are relaxed, in that neither $\Delta_k^p = \Delta_k^m$ nor $D_k \subset D$ generally hold. The poll step evaluates points lying in a frame (analogous to the poll set in GPS), constructed differently than GPS. The MADS frame is defined as follows (see [11]).

Definition 3.1.1. *At each iteration k, the MADS frame is defined to be the set*

$$P_k = \{x_k + \Delta_k^m d : d \in D_k\} \subset M_k \tag{3.9}$$

where D_k is a positive spanning set such that $0 \notin D_k$ and for each $d \in D_k$ the following conditions must be met [11]:

1. *d can be written as a nonnegative integer combination of the directions in D : $d = Du$ for some vector $u \in \mathbb{N}^{n_{D_k}}$ that may depend on the iteration number k,*
2. *the distance from the frame center x_k to a frame point $x_k + \Delta_k^m d \in P_k$ is bounded above by a constant times the poll size parameter:*

$$\Delta_k^m \parallel d \parallel \leq \Delta_k^p \max \{\parallel d' \parallel : d' \in D\},$$

3. *limits of the normalized sets $D_k = \left\{ \frac{d}{\|d\|} : d \in D_k \right\}$ are positive spanning sets.*

A General MADS Algorithm

- INITIALIZATION: Let $x_0 \in \Omega$, $\Delta_0^m \leq \Delta_0^p$, D, G, τ, w^-, and w^+ satisfy the requirements of a MADS frame set given in Definition 3.1.1. Set the iteration counter $k \leftarrow 0$.
- SEARCH AND POLL STEP: Perform the SEARCH and possibly the POLL steps (or part of them) until an improved mesh point x_{k+1} is found on the mesh M_k (where M_k is defined as for GPS in Equation (3.4) in Section 3.1.2).
  - OPTIONAL SEARCH: Evaluate f_Ω on a finite subset of trial points on the mesh M_k.
  - LOCAL POLL: Evaluate f_Ω on the frame P_k (where P_k is as given in Equation (3.9) in Section 3.1.2).
- PARAMETER UPDATE: Update Δ_{k+1}^m according to Equations (3.6) and (3.7) and Δ_{k+1}^p so that Equation (3.8) is satisfied. Set $k \leftarrow k+1$ and go back to the SEARCH AND POLL step.

Figure 3.3: A General MADS Algorithm [11]

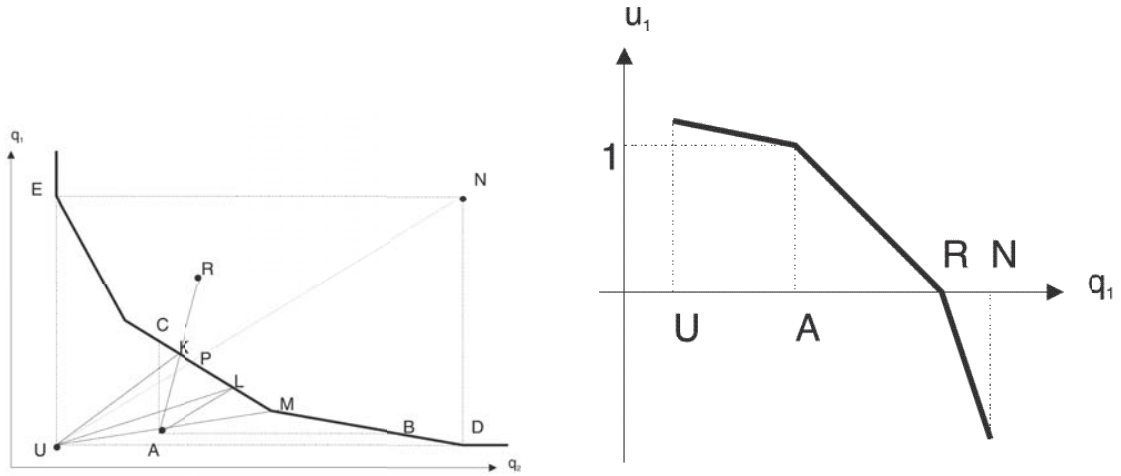
The general MADS algorithm, as developed by Audet and Dennis, is shown in Figure 3.3. This algorithm is extended to the mixed variable case in [5] and in Section 3.3.1. Extension to the stochastic case is accomplished via ranking and selection in a manner similar to MGPS-RS. The extended algorithm, mixed variable mesh adaptive direct search with ranking and selection (MVMADS-RS), is shown in Figure 3.4.

3.1.4 Interactive Specification of Aspiration/Reservation Levels and Scalarization Functions. As shown in Figure 3.5(a), points on the Pareto front can be found by varying the relative importance, *i.e.*, trade-off coefficients or weights, of the distance to a given point. Using the utopia point **U**, any point between points **D** and **E** can be found. (Points **D** and **E** correspond to Pareto optimal solutions found by using the utopia point in one dimension and the nadir point in the other, *i.e.*, the best possible solution for one objective, but not for the other.) By using aspiration point **A** and varying the weights or slope of the ray emanating from it, points between **B** and **C** can be found. There are many methods for determining which ray to use [92]. This particular method uses the reservation point **R** as the second point

A General MVMADS-RS Algorithm

- INITIALIZATION: Let $X_0 \in \Omega$, $\Delta_k^p \geq \Delta_k^m > 0$, $\xi > 0$, $\alpha_0 \in (0,1)$ and $\delta_0 > 0$. Set the iteration and R&S counters $k = 0$ and $r = 0$ respectively.
- POLL STEP: Set extended poll trigger $\xi_k \geq \xi$. Use R&S procedure $RS(P_k(X_k) \bigcup \mathcal{N}(X_k), \alpha_r, \delta_r)$ to return the estimated best solution $\hat{Y}$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$. If $\hat{Y} \neq X_k$, the step is *successful*, update $X_{k+1} = \hat{Y}$, $\Delta_{k+1}^p \geq \Delta_k^p$ according to Equations (3.6) and (3.7), $\Delta_{k+1}^m \geq \Delta_k^m$ so that Equation (3.8) is satisfied, and $k = k+1$ and return to POLL STEP. Otherwise, proceed to EXTENDED POLL STEP.
- EXTENDED POLL STEP: For each discrete neighbor $Y \in \mathcal{N}(X_k)$ that satisfies the extended poll trigger condition $\overline{F}(Y) < \overline{F}(X_k) + \xi_k$, set $j = 1$ and $Y_k^j = Y$ and do the following.
  - Use R&S procedure $RS(P_k(Y_k^j), \alpha_r, \delta_r)$ to return the estimated best solution $\hat{Y}$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$. If $\hat{Y} \neq Y_k^j$, set $Y_k^{j+1} = \hat{Y}$ and $j = j + 1$ and repeat this step. Otherwise, set $Z_k = Y_k^j$ and go to the next step.
  - Use R&S procedure $RS(X_k \bigcup Z_k), \alpha_r, \delta_r)$ to return the estimated best solution $\hat{Y}$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$. If $\hat{Y} = Z_k$, the step is successful, update $X_{k+1} = \hat{Y}$, $\Delta_{k+1}^p \geq \Delta_k^p$ according to Equations (3.6) and (3.7), $\Delta_{k+1}^m \geq \Delta_k^m$ so that Equation (3.8) is satisfied, and $k = k + 1$ and return to the POLL STEP. Otherwise, repeat the EXTENDED POLL STEP for another discrete neighbor that satisfies the extended poll trigger condition. If no such discrete neighbors remain in $\mathcal{N}(X_k)$, set $X_{k+1} = X_k$, $\Delta_{k+1}^p < \Delta_k^p$ according to Equations (3.6) and (3.7), $\Delta_{k+1}^m < \Delta_k^m$ so that Equation (3.8) is satisfied, and $k = k + 1$ and return to the POLL STEP.

Figure 3.4: The Mixed-variable MADS with Ranking and Selection (MVMADS-RS) Algorithm



(a) Pareto solutions corresponding to different component achievement functions (Figure 3 in [96])

(b) Component Achievement Functions for Minimized Criteria (Figure 4 in [96])

Figure 3.5: Graphical illustrations of functions used for analysis of Pareto optimal solutions

in determining the direction of the ray [96].

This technique is based on the assumption that the decision maker has an idea of what is desired for each objective, as well as what minimum, or maximum, values are acceptable. These values are referred to as the aspiration and reservation values, respectively; *i.e.*, points **A** and **R** discussed previously and shown in Figure 3.5(a). These values are then used inside of an achievement scalarization function of the form shown graphically in Figure 3.5(b) and given by

$$S(q, \bar{q}, \underline{q}) = \min_{1 \le i \le J} u_i(q_i, \bar{q}_i, \underline{q}_i) + \epsilon \sum_{i=1}^{J} u_i(q_i, \bar{q}_i, \underline{q}_i). \tag{3.10}$$

The function u_i is called a *component achievement function*, *i.e.*, strictly monotone function of the objective function vector components $q_i = f_i(x)$, $i = 1, 2, \ldots, n$, given by

$$u_i(q_i, \bar{q}_i, \underline{q}_i) = \begin{cases} \alpha_i w_i(\bar{q}_i - q_i) + 1, & q_i < \bar{q}_i \\ w_i(\bar{q}_i - q_i) + 1, & \bar{q}_i \le q_i \le \underline{q}_i \\ \beta_i w_i(\underline{q}_i - q_i), & \underline{q}_i < q_i, \end{cases} \tag{3.11}$$

where $\bar{q}_i$ and $\underline{q}_i$ are the aspiration and reservation levels, respectively, for objective i, $w_i = 1/\left(\underline{q}_i - \bar{q}_i\right)$, and α_i, β_i, $i = 1, 2, \ldots, J$, are given parameters that are set in such a way that u_i takes the values at the utopia, aspiration, reservation, and nadir points, given by

$$u_i\left(q_i^U, \cdot\right) = 1 + \bar{\beta}, \qquad u_i\left(\bar{q}_i, \cdot\right) = 1, \qquad u_i\left(\underline{q}_i, \cdot\right) = 0, \qquad u_i\left(q_i^R, \cdot\right) = -\bar{\eta},$$

respectively, and $\bar{\beta}$ and $\bar{\eta}$ are given positive constants, typically set to 0.1 and 10, respectively [96]. The maximization of Equation (3.10) provides proper Pareto optimal solutions nearest the aspiration level (see point **K** in Figure 3.5(a)).

3.1.5 Multi-Objective Ranking and Selection. Lee *et al.* [83] propose a performance index to measure the degree that a point is dominated in the Pareto sense when the objective function evaluations are subject to noise. They then use this measure in the Multi-objective Optimal Computing Budget Allocation (MOCBA) algorithm to determine a Pareto optimal set for multi-objective simulation-based optimization problems [86]. Given a set of designs $i = 1, 2, \ldots, n$, evaluated by J performance measures μ_{ij}, $j = 1, 2, \ldots, J$, through some simulation (or other estimation method), μ_{ij} is a random variable whose Bayesian posterior distribution can be derived based on its prior distribution and the simulation output [83]. Define a performance index

$$\psi_i = \prod_{k=1, k\neq i}^{n} \left[1 - P\left(\mu_k \prec \mu_i\right)\right], \tag{3.12}$$

where $P\left(\mu_k \prec \mu_i\right)$ represents the probability that design k dominates design i. Under the assumption that the performance measures are independent and follow continuous distributions such that

$$P\left(\mu_k \prec \mu_i\right) = \prod_{j=1}^{J} P\left(\mu_{kj} \leq \mu_{ij}\right),$$

ψ_i measures the probability that design i is not dominated by any other designs and can be used inside of a ranking and selection framework to find the set of non-dominated points rather than a single best point. When approximating a Pareto set, two types of error exist: Type I (e_1), the probability that at least one design left out of the efficient set is non-dominated, and Type II (e_2), the probability that at least one design in the efficient set is dominated. As these errors both approach zero, the experimental Pareto set approaches the true[1] Pareto set. It is shown that these errors are bounded by the approximated errors ae_1 given by

$$e_1 \leq ae_1 = \sum_{i \in \overline{S}_p} \psi_i$$

and ae_2 given by

$$e_2 \leq ae_2 = \sum_{i \in S_p} (1 - \psi_i),$$

respectively, and that the observed Pareto set determined by MOCBA approaches the true Pareto set asymptotically with probability 1 [84, 85].

In each iteration of MGPS-RS or MADS, only a finite number of points are evaluated by the ranking and selection procedure. Thus, this type of ranking and selection method could be substituted for the single objective ranking and selection method inside the MGPS-RS or MADS algorithms to develop multi-objective versions. Alternatively, the performance measure ψ could be substituted for the objective function value inside of a traditional ranking and selection procedure. In this case, the convergence results from [84, 85] would have to be reverified.

[1] In practice, the complete true Pareto set is not found. In this context, true Pareto set refers to an approximation containing only efficient points in which no efficient points were considered but then not selected as such.

3.2 Stochastic Multi-Objective Mesh Adaptive Direct Search (SMOMADS)

A new two-stage algorithm which uses the methods of Section 3.1 to numerically solve mixed variable stochastic multi-objective optimization problems is now introduced. In the first stage, a convex combination of objectives, via scalarization functions and aspiration/reservation levels of the decision maker, is used to determine an approximation of the Pareto front in a region of interest. For each pair of aspiration and reservation levels, MGPS-RS or MVMADS-RS can be used to generate a single Pareto point. (Extension of MADS to the mixed-variable case—MVMADS—is discussed in Sections 3.2.1.4 and 3.3.1 and extension to the stochastic case—MVMADS-RS—is discussed in Section 3.3.2.) However, since convexity of the actual Pareto frontier is not assumed, some regions in the Pareto frontier may not contain any of the generated points. In the second stage, the single-objective ranking and selection routine inside of MGPS-RS is replaced with MOCBA, so that the discrete points in the mesh can be evaluated with respect to multiple objectives.[2] A graphical representation is shown in Figure 3.6 and descriptions of each step follow.

3.2.1 Stage 1. Stage 1 of the method consists of aspiration and reservation level analysis in order to represent the original problem as one with a single objective that can then be solved using MGPS-RS or MVMADS-RS.

3.2.1.1 Aspiration and Reservation Level Analysis. As discussed in Section 3.1.4, the multiple objectives are combined into a single objective problem of the form shown in Equation (3.10). Each *subproblem*, or choice of a specific pairing of aspiration and reservation levels, produces an approximate Pareto point. There are many ways to produce such aspiration/reservation level pairings. (In this algorithm, the problem of determining these pairings is referred to as the *master problem*.) His-

[2]The theoretical convergence properties developed in Section 3.3.3 do not depend on stage 2 of the algorithm. It was added to better determine non-convex Pareto frontiers. As the algorithm was implemented and tested, the good performance of Stage 1 made Stage 2 unnecessary for the problems tested. Since it may still be helpful for many problems, Stage 2 remains part of the algorithm.

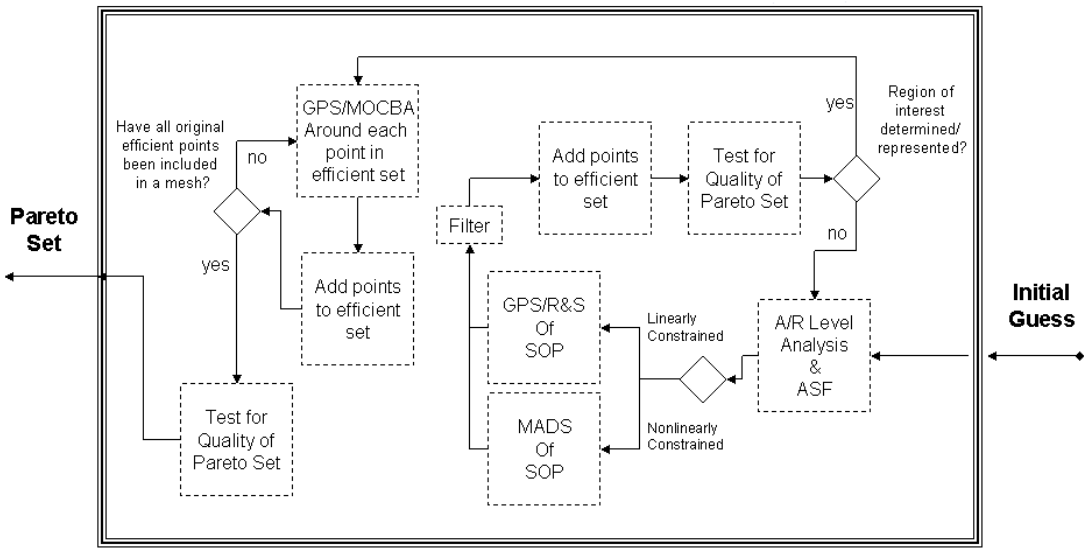


Figure 3.6: Stochastic Multi-Objective Mesh Adaptive Direct Search (SMOMADS)

torically, in interactive specification of aspiration and reservation levels, a decision maker was actively involved in choosing these points [62]. However, if this interaction is not possible or if the decision maker has only specified a range of values for aspiration and reservation levels, some other method must be used. In the case where a range of values has been specified, the problem is that of determining an approximation to the Pareto frontier within a region of interest. Such a problem is similar to that of approximating a response surface with aspiration and reservation levels as the decision variables. Thus, experimental design methods apply.

3.2.1.2 Example. Given a problem of the form

$$\min_{x \in \Omega}(f_1(x), f_2(x))$$

where Ω represents some feasible region and that the (as yet unknown) Pareto front is as shown by the curve in Figure 3.7. The utopia and nadir points are determined to be (1,3) and (6,8), and depicted by points **U** and **R**, respectively, in the figure.

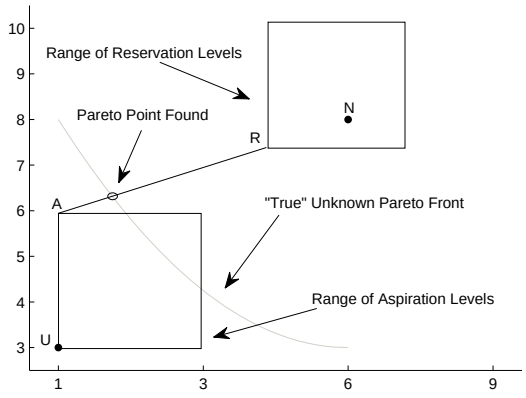


Figure 3.7: Graphical Example of Aspiration/Reservation Level Analysis

After discussion with the decision maker, it is decided that the algorithm should examine aspiration levels between 1 and 3 for objective 1 and between 3 and 6 for objective 2 and examine reservation levels between 4 and 7 for objective 1 and between 7 and 10 for objective 2. (This is depicted by the rectangles in Figure 3.7.) A central composite design with 4 factors (aspiration and reservation levels for 2 objectives) is constructed, resulting in 36 test points, *i.e.*, 36 pairs of aspiration and reservation levels. One of these pairings is shown by the points **A** and **R** in Figure 3.7. This pair is used to determine the component achievement functions using Equation (3.11), which become

$$u_1(q_1, 1, 4) = \begin{cases} \alpha_1 w_1(1 - q_1) + 1, & q_1 < 1 \\ w_1(1 - q_1) + 1, & 1 \leq q_1 \leq 4 \\ \beta_1 w_1(4 - q_1), & 4 < q_1 \end{cases}$$

$$u_2(q_2, 6, 7) = \begin{cases} \alpha_2 w_2(6 - q_2) + 1, & q_2 < 6 \\ w_2(6 - q_2) + 1, & 6 \leq q_2 \leq 7 \\ \beta_2 w_2(7 - q_2), & 7 < q_2. \end{cases}$$

These equations are used by Equation (3.10), the single objective to be solved by the subproblem.

3.2.1.3 MGPS-RS for Problems with Linear Constraints. This step uses the NOMADm [3] implementation of MGPS-RS to solve each single objective subproblem of the form discussed in Section 3.1.4. MGPS-RS is discussed in detail in Section 3.1.2 and has been shown to have strong convergence properties. (See Section 3.3.3.1 and [137].)

3.2.1.4 MV-MADS for Problems with Nonlinear Constraints. Similarly, this step of SMOMADS uses the NOMADm implementation of MVMADS-RS [3] to solve each single objective subproblem of the form discussed in Section 3.1.4. MADS is discussed in detail in Section 3.1.3 and has also been shown to have strong convergence properties for the deterministic case [11]. (See Section 3.3.1 and [5] for MV-MADS.) Conjecture of convergence results for the stochastic case (MVMADS-RS) are discussed in Section 3.3.10.

3.2.1.5 Adding Points to the Efficient Set. The algorithm is designed so that each subproblem should produce an efficient point. In fact, this is always the case for deterministic problems (see Lemma 3.3.11). In stochastic problems, as the number of iterations of the single objective solver is allowed to approach infinity, the solution converges to an efficient point with probability one (see Theorem 3.3.12 and Conjecture 3.3.10). However, in practice, since the number of iterations is finite, the addition of dominated points is possible. Therefore, points are checked for Pareto dominance before they are allowed to enter the efficient set. If a point is dominated, it is "filtered" and does not enter the efficient set. Ideally, the filter should also check to see if the new point dominates other points in the current efficient set. This feature is suggested for follow-on research in Chapter 6.2.3.3.

3.2.1.6 Tests for Quality of the Pareto Set. An exact Pareto set often has an infinite number of efficient points. Since multi-objective solvers provide only an approximate Pareto set, the quality of the approximation is of particular interest. Relatively few papers in the literature focus on quality metrics for Pareto

set approximations and most make the assumption that the true set is known *a priori*. Since no such assumption is made here, the quality metrics introduced by Wu and Azarm [163] are used to assess the quality of the Pareto set, because their metrics measure the quality (accuracy, spread, cluster, etc.) without any knowledge of the true Pareto set. Specific metrics, as introduced by Wu and Azarm [163], are described below:

1. **Scaled Objective Space.** All the quality metrics used in this research are based on the existence of "bad" objective function values $\{f_1^b, \ldots, f_J^b\}$ that are worse than all points in the Pareto set, and "good" objective function values $\{f_1^g, \ldots, f_J^g\}$ that are better than all points in the Pareto set. The utopia and nadir points can be used for these values, other points, which define a smaller region, can also be used, as determined by the user. All the metrics require that Pareto optimal points be scaled using

$$\overline{f}_j(x_k) = \frac{f_j(x_k) - f_j^g}{f_j^b - f_j^g}, \tag{3.13}$$

   where $f_j(x_k)$ is the value of the j-th objective at the point x_k, $\overline{f}_j(x_k)$ is the scaled value of the j-th objective at the point x_k, and f_j^g and f_j^b are the good and bad values, respectively, of the j-th objective function. The metrics that follow are heavily dependent on choice of f^g and f^b points. Thus these metrics are better suited to relative comparisons between sets, *e.g.*, measuring how the quality of the set improves as points are added, rather than giving an absolute measure of the quality of a particular set.

2. **Hyperarea Difference.** The *hyperarea difference* ($HD(P)$) provides a quantitative means to evaluate the difference between the size of the scaled objective space dominated by an approximate Pareto set and that of the space dominated by the true set itself (which is equal to 1 when scaled). Although the actual Pareto set is usually unknown, it is still possible to instead compare two experimental sets with respect to how much worse they are than the true Pareto

in the scaled objective space. The salient point is that, though the sets are measured with respect to the true set, the comparison and inferences are of the two experimental sets in relation to each other. Given an observed Pareto optimal set, scaled appropriately by Equation (3.13), the hyperarea difference can be calculated as

$$HD(P) = 1 - \left\{ \sum_{r=1}^{\overline{np}} \left\{ (-1)^{r+1} \left[\sum_{k_1=1}^{\overline{np}-r+1} \cdots \sum_{k_l=k_{l-1}+1}^{\overline{np}-(r-l+1)+1} \cdots \sum_{k_r=k_r-1}^{\overline{np}} \prod_{i=1}^{m} \left[1 - \max_{j=1}^{r} \left(\overline{f}_i(x_{k_j}) \right) \right] \right] \right\} \right\}$$

where $\overline{np}$ is the number of observed Pareto optimal points. The hyperarea metric becomes difficult to calculate as the number of points in the efficient set becomes large. For this reason, this metric was not calculated. However, it could be useful in the determination of termination criteria, as described in Section 4.4.2.

3. **Pareto Spread.** The *Pareto spread* measures how widely the experimental Pareto optimal set spreads over the objective space. The *overall Pareto spread* ($OS(P)$) considers all objectives together and is defined as the volume ratio of two hyper-rectangles. One rectangle is defined by f^b and f^g for each objective and the other is defined by the extreme points in the experimental Pareto optimal set. The overall Pareto spread can calculated as

$$OS(P) = \prod_{i=1}^{m} \left| \max \left\{ \overline{f}_i(x_k) : k = 1, 2, \ldots, \overline{np} \right\} - \min \left\{ \overline{f}_i(x_k) : k = 1, 2, \ldots, \overline{np} \right\} \right|.$$

Similarly, the *k-th objective Pareto spread*

$$OS_k(P) = \left| \max \left\{ \overline{f_k}(x_i) : i = 1, 2, \ldots, \overline{np} \right\} - \min \left\{ \overline{f_k}(x_i) : i = 1, 2, \ldots, \overline{np} \right\} \right|$$

is a measure of how the experimental Pareto optimal set spreads over the objective space when each objective is considered individually.

4. **Number of Distinct Choices.** When comparing two Pareto sets, the one with more points may appear to be better. However, this may not be the case. If the points in a Pareto set P are too close together, according to some distance metric, they may then be indistinguishable from each other for practical design purposes. The *number of distinct choices* ($NDC_\mu(P)$) is a measure of the number of Pareto points that are distinguishable (*i.e.*, far enough away) from each other. It is calculated as follows. A quantity μ is given by the decision maker which divides the m-dimensional objective space into $1/\mu^m$ small grids. The indicator variable

$$NT_\mu(q, P) = \begin{cases} 1, & \exists\ p_k \in P \quad p_k \in T_\mu(q) \\ 0, & \forall\ p_k \in P \quad p_k \notin T_\mu(q) \end{cases}$$

is used to indicate whether a particular region $T_\mu(q)$ contains any point p_k in the Pareto set. The number of distinct choices can then be calculated by summing the number of regions that contain at least one point,

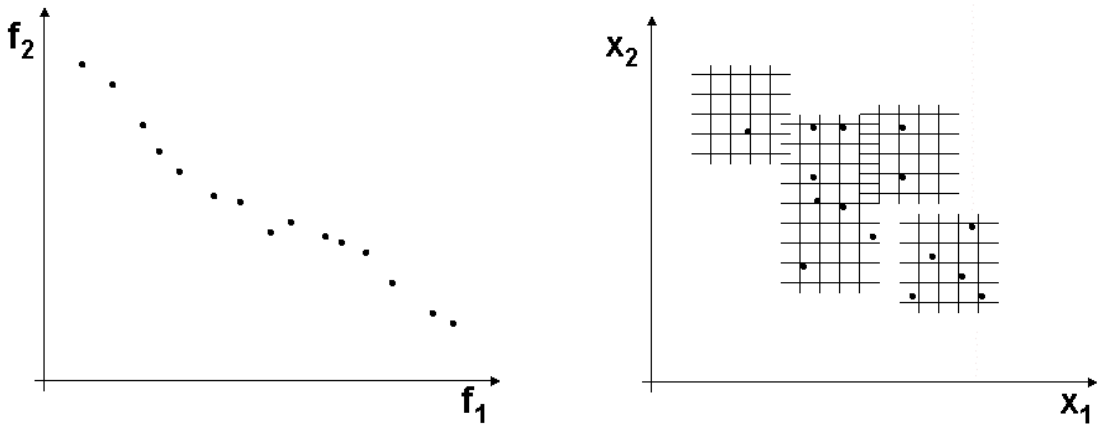
$$NDC_\mu(P) = \sum_{l_m=0}^{v-1} \cdots \sum_{l_2=0}^{v-1} \sum_{l_1=0}^{v-1} NT_\mu(q, P)$$

where $q = (q_1, q_2, \ldots, q_m)$ with $q_i = \dfrac{l_i}{v}$.

5. **Cluster.** Like the previous metric, the *cluster metric* $CL_\mu(P)$ quantifies how closely spaced together and, as can be seen in

$$CL_\mu(P) = \frac{N(P)}{NDC_\mu(P)},$$

it is the ratio of the number of experimental Pareto solutions $N(P)$ to the number of distinct choices.



(a) Objective Space (b) Design Space Showing Mesh

Figure 3.8: Notional approximated Pareto set and mesh for a two-objective problem with two design variables

3.2.2 Stage 2. Though interactive specification of aspiration/reservation levels and scalarization functions of the type discussed in Section 3.1.4 is capable of finding Pareto points in nonconvex regions of the Pareto front, it is not guaranteed to find all points. It is possible that some efficient points may not be found. Therefore, a second (optional) stage is added for those cases in which missing points may pose a particular problem.[3] This stage does not effect the convergence theory. In this stage, a discrete mesh (similar to that used for MGPS-RS and MADS) around the current efficient points is generated (see Section 3.1.2) and then a multi-objective ranking and selection algorithm is used to check for new efficient points on the mesh, which is similar to the frame used by MADS given in Equation (3.4), except that S_0 is replaced by S_e, the set of efficient points found in stage 1. A graphical representation of a notional problem is shown in Figure 3.8.

3.2.2.1 Multi-Objective Ranking and Selection. A version of Multi-objective Optimal Computing Budget Allocation algorithm (MOCBA), developed by Lee *et al.* is used to check for new efficient points on the mesh. As discussed in Section

[3]In this research, the algorithm performed so well on the test problems that the second stage was not necessary; however, it is suggested as follow-on research in Section 6.2.2.

3.1.5, MOCBA has been used successfully for multi-objective ranking and selection problems [83,86] and the observed Pareto set determined by MOCBA approaches the true Pareto set asymptotically with probability 1 [85].

3.3 Convergence Results

Since existing convergence results for MADS apply only to NLP problems, the first part of this subsection provides a convergence theory for an extension of MADS to mixed variable problems, called *MV-MADS*. The extension is very straightforward, so that the algorithm is almost identical to mixed variable pattern search, except for the traditional differences between GPS and MADS with respect to poll directions and and mesh parameters. This work is based on the joint work done in [5], and the results presented here match those of [5], unless otherwise noted.

3.3.1 Convergence Results for Mixed-Variable MADS (MV-MADS). Given a constrained optimization problem with objective function $f : \Omega \rightarrow \mathbb{R}$, where Ω represents the feasible region, let $f_\Omega : \mathbb{R}^n \rightarrow \mathbb{R}$ be defined by $f_\Omega = f + \psi$, where $\psi(x) = 0$ if $x \in \Omega$, and $+\infty$ otherwise, and assume the following:

A1. An initial point x_0 with $f_\Omega(x_0) < \infty$ is available.

A2. All iterates $\{x_k\}$ generated by MV-MADS lie in a compact set.

A3. The set of discrete neighbors $\mathcal{N}(x_k)$ lies on the mesh M_k.

Under these assumptions, the following results are obtained by proofs that are identical to those found in Audet and Dennis [9] and Abramson [2] for mixed variable GPS:

- $\liminf_{k\rightarrow+\infty} \Delta_k^p = \liminf_{k\rightarrow+\infty} \Delta_k^m = 0$;
- there exists a *refining subsequence* $\{x_k\}_{k\in K}$ of minimal frame centers for which there are limit points $\hat{x} = \lim_{k\in K} x_k$, $\hat{y} = \lim_{k\in K} y_k$, and $\hat{z} = (\hat{z}^c, \hat{y}^d) = \lim_{k\in K} z_k$, where each $z_k \in \Omega$ is the endpoint of the EXTENDED POLL step initiated at $y_k \in \mathcal{N}(x_k)$, and $\lim_{k\in K} \Delta_k^p = 0$.

Some of the results that follow require the additional assumption that $\hat{y} \in \mathcal{N}(\hat{x})$. Additionally, note the following properties from Clarke [30]:

- Any convex set is regular at each of its points.
- Both $T_\Omega^{Co}(x)$ and $T_\Omega^{Cl}(x)$ are closed, and both $T_\Omega^{Cl}(x)$ and $T_\Omega^{H}(x)$ are convex.
- $T_\Omega^{H}(x) \subseteq T_\Omega^{Cl}(x) \subseteq T_\Omega^{Co}(x)$.
- Rockafellar [124] showed that, if $T_\Omega^{H}(x)$ is nonempty, $T_\Omega^{H}(x) = \text{int}(T_\Omega^{Cl}(x))$, and therefore, $T_\Omega^{Cl}(x) = \text{cl}(T_\Omega^{H}(x))$.

A generalization [74] of the Clarke [30] directional derivative, in which function evaluations are restricted to points in the domain, is needed. The generalized directional derivative of a locally Lipschitz function f at $x \in \Omega$ in the direction $v \in \mathbb{R}^n$ is defined by

$$f^\circ(x;v) := \limsup_{\substack{y \to x,\ y \in \Omega \\ t \downarrow 0,\ y+tv \in \Omega}} \frac{f(y+tv)-f(y)}{t}. \tag{3.14}$$

The following four definitions [30, 74, 124] are needed. They have been adapted to the context of this particular problem, where only a subset of the variables are continuous. The standard definitions follow when all variables are continuous, *i.e.*, when $n^d = 0$ and $x = x^c$.

Definition 3.3.1. *A vector $v \in \mathbb{R}^{n^c}$ is said to be a* hypertangent vector *to the continuous variables of the set Ω at the point $x = (x^c, x^d) \in \Omega$ if there exists a scalar $\varepsilon > 0$ such that*

$$(y + tw, x^d) \in \Omega \quad \textit{for all } y \in B_\varepsilon(x^c) \textit{ with } (y, x^d) \in \Omega,\ w \in B_\varepsilon(v) \quad \textit{and} \quad 0 < t < \varepsilon.$$

The set $T_\Omega^H(x)$ of all hypertangent vectors to Ω at x is called the hypertangent cone to Ω at x.

Definition 3.3.2. *A vector $v \in \mathbb{R}^{n^c}$ is said to be a* Clarke tangent vector *to the continuous variables of the set Ω at the point $x = (x^c, x^d) \in \mathrm{cl}(\Omega)$ if for every sequence $\{y_k\}$ that converges to x^c with $(y_k, x^d) \in \Omega$ and for every sequence of positive real numbers $\{t_k\}$ converging to zero, there exists a sequence of vectors $\{w_k\}$ converging to v such that $(y_k + t_k w_k, x^d) \in \Omega$. The set $T_\Omega^{Cl}(x)$ of all Clarke tangent vectors to Ω at x is called the* Clarke tangent cone *to Ω at x.*

Definition 3.3.3. *A vector $v \in \mathbb{R}^{n^c}$ is said to be a* tangent vector *to the continuous variables of the set Ω at the point $x = (x^c, x^d) \in \mathrm{cl}(\Omega)$ if there exists a sequence $\{y_k\}$ that converges to x^c with $(y_k, x^d) \in \Omega$ and a sequence of positive real numbers $\{\lambda_k\}$ for which $v = \lim_k \lambda_k(y_k - x^c)$. The set $T_\Omega^{Co}(x)$ of all tangent vectors to Ω at x is called the* contingent cone *to Ω at x.*

Definition 3.3.4. *The set Ω is said to be* regular *at x if $T_\Omega^{Cl}(x) = T_\Omega^{Co}(x)$.*

The results of this section make use of a generalization [74] of the Clarke [30] directional derivative, in which function evaluations are restricted to points in the domain. Furthermore, the notions of generalized directional derivatives and gradient are restricted to the subspace of continuous variables. The generalized directional derivative of a locally Lipschitz function f at $x = (x^c, x^d) \in \Omega$ in the direction $v \in \mathbb{R}^{n^c}$ is defined by

$$f^\circ(x; v) := \limsup_{\substack{y \to x^c,\ (y, x^d) \in \Omega \\ t \downarrow 0,\ (y + tv, x^d) \in \Omega}} \frac{f(y + tv, x^d) - f(y, x^d)}{t}. \tag{3.15}$$

Furthermore, it is shown in [11] that if $T_\Omega^H(x)$ is not empty and $v \in T_\Omega^{Cl}(x)$, then

$$f^\circ(x; v) = \lim_{\substack{u \to v, \\ u \in T_\Omega^H(x)}} f^\circ(x; u). \tag{3.16}$$

Other notions regarding derivatives are generalized as follows. Denote by $\nabla f(x) \in \mathbb{R}^{n^c}$ and $\partial f(x) \subseteq \mathbb{R}^{n^c}$, respectively, the gradient and generalized gradient of the func-

tion f at $x = (x^c, x^d) \in \Omega$ with respect to the continuous variables x^c while holding the categorical variables x^d constant. In particular, the generalized gradient of f at x^* [30] with respect to the continuous variables is defined by

$$\partial f(x) := \left\{ s \in \mathbb{R}^{n^c} : f^\circ(x; v) \geq v^T s \text{ for all } v \in \mathbb{R}^{n^c} \right\}.$$

The function f is said to be *strictly differentiable* at x with respect to the continuous variables if the generalized gradient of f with respect to the continuous variables at x is a singleton; *i.e.*, $\partial f(x) = \{\nabla f(x)\}$.

The final definition, which is adapted from the original MADS algorithm in [11] for the mixed variable case, provides some terminology for stationarity in the nonsmooth case.

Definition 3.3.5. *Let f be Lipschitz near $x^* \in \Omega$. Then x^* is said to be a* Clarke, or contingent, stationary point *of f over Ω with respect to the continuous variables if $f^\circ(x^*; v) \geq 0$ for every direction v in the Clarke tangent cone to Ω at x^*, or contingent cone to Ω at x^*, respectively.*

In addition, x^* is said to be a Clarke, or contingent, KKT stationary point *of f over Ω if $-\nabla f(x^*)$ exists and belongs to the polar of the Clarke tangent cone to Ω at x^*, or contingent cone to Ω at x^*, respectively.*

If $\Omega^c(x^d) = \mathbb{R}^{n^c}$ or x^{*c} lies in the relative interior of $\Omega^c(x^d)$, then a stationary point as described by Definition 3.3.5 meets the condition that $f^\circ(x^*; v) \geq 0$ for all $v \in \mathbb{R}^{n^c}$. This is equivalent to $0 \in \partial f(x^*)$.

Main convergence results for the MVMADS algorithm consist of four theorems, all of which are generalizations of similar results from MADS [11] or mixed variable pattern search [2, 6, 9]. The first result establishes a notion of directional stationarity at certain limit points, and the second ensures local optimality with respect to the set of discrete neighbors. The remaining two results establish Clarke-based stationarity in a mixed variable sense.

Theorem 3.3.6. *Let $\hat{w}$ be the limit point of a refining subsequence or the associated subsequence of* EXTENDED POLL *endpoints, and let v be a refining direction in the hypertangent cone $T_X^H(\hat{w})$. If f is Lipschitz at $\hat{w}$ with respect to the continuous variables, then $f^\circ(\hat{w};(v,0)) \geq 0$.*

Proof. Let K be an indexing set of $\{w_k\}_{k\in K}$, a refining subsequence converging to $\hat{w} = (\hat{w}^c, \hat{w}^d)$. Without any loss of generality, assume that $w_k = (w_k^c, \hat{w}^d)$ for all $k \in K$. In accordance with Definition 3.2 in [11], let $v = \lim_{k\in L} \frac{d_k}{\|d_k\|} \in T_\Omega^H(\hat{w})$ be a refining direction for $\hat{w}$, where $d_k \in D_k$ for all $k \in L$ and L is some subset of K.

Because w_k converges to $\hat{w}$ and by the definition of the MADS poll set, $\Delta_k^m\|d_k\|$ is bounded above by $\Delta_p^k \max\{\|d'\| : d' \in D\}$ where D is a finite set of directions and Δ_p^k converges to zero, it follows that $\Delta_k^m\|d_k\|$ also converges to zero. Thus, it also follows from Equation (3.15) that:

$$\begin{aligned} f^\circ(\hat{w};v) &= \limsup_{\substack{y\to\hat{w}^c,\ (y,w^d)\in\Omega \\ t\downarrow 0,\ (y+tu,w^d)\in\Omega \\ u\to v,\, u\in T_\Omega^H(\hat{w})}} \frac{f\left(y+tu,\hat{w}^d\right) - f(y,\hat{w}^d)}{t} \\ &\geq \limsup_{k\in L} \frac{f\left(w_k^c + \Delta_k^m\|d_k\|\frac{d_k}{\|d_k\|},\hat{w}^d\right) - f(w_k^c,\hat{w}^d)}{\Delta_k^m\|d_k\|} \\ &= \limsup_{k\in L} \frac{f\left(w_k+\Delta_k^m d_k,\hat{w}^d\right) - f(w_k)}{\Delta_k^m\|d_k\|} \ \geq\ 0. \end{aligned}$$

The last inequality holds because $(w_k+\Delta_k^m d_k, \hat{w}^d) \in \Omega$ and $f(w_k+\Delta_k^m d_k, \hat{w}^d) \geq f(w_k)$ (since w_k is a minimal frame center) for all sufficiently large $k \in L$. □

The next result gives sufficient conditions under which $\hat{x}$ is a local minimizer with respect to its discrete neighbors.

Theorem 3.3.7. *If f is lower semi-continuous at $\hat{x}$ and upper semi-continuous at $\hat{y} \in \mathcal{N}(\hat{x})$ with respect to the continuous variables, then $f(\hat{x}) \leq f(\hat{y})$.*

Proof. Since $k \in K$ ensures that $\{x_k\}_{k\in K}$ are minimal frame centers, it follows that $f(x_k) \leq f(y_k)$ for all $k \in K$. By the assumptions of lower and upper semi-continuity on f and the definitions of $\hat{x}$ and $\hat{y}$, it also follows that $f(\hat{x}) \leq \lim_{k\in K} f(x_k) \leq \lim_{k\in K} f(y_k) = f(\hat{y})$. □

The next theorem lists conditions that ensure that $\hat{x}$ satisfies certain stationary conditions, under various smoothness requirements.

Theorem 3.3.8. *Assume that $T_\Omega^H(\hat{x}) \neq \emptyset$ and the set of refining directions is asymptotically dense in $T_\Omega^H(\hat{x})$.*

1. *If f is Lipschitz near $\hat{x}$ with respect to the continuous variables, then $\hat{x}$ is a Clarke stationary point of f on Ω with respect to the continuous variables.*
2. *If f is strictly differentiable at $\hat{x}$ with respect to the continuous variables, then $\hat{x}$ is a Clarke KKT stationary point of f on Ω with respect to the continuous variables.*

Furthermore, if Ω is regular at $\hat{x}$, then the following hold:

1. *If f is Lipschitz near $\hat{x}$ with respect to the continuous variables, then $\hat{x}$ is a contingent stationary point of f on Ω with respect to the continuous variables.*
2. *If f is strictly differentiable at $\hat{x}$ with respect to the continuous variables, then $\hat{x}$ is a contingent KKT stationary point of f on Ω.*

Proof. First, Rockafellar [124] showed that if the hypertangent cone is not empty at $\hat{x}$, then $T_\Omega^{Cl}(\hat{x}) = \mathrm{cl}(T_\Omega^H(\hat{x}))$. Since the set S of refining directions for f at $\hat{x}$ is a dense subset of $T_\Omega^H(\hat{x})$, S is also a dense subset of $T_\Omega^{Cl}(\hat{x})$. Thus, any vector $v \in T_\Omega^{Cl}(\hat{x})$ can be expressed as the limit of directions in S, and the first result follows directly from (3.16) and Theorem 3.3.6.

Strict differentiability ensures the existence of $\nabla f(\hat{x})$ and that $\nabla f(\hat{x})^T v = f^\circ(\hat{x}; v)$ for all $v \in T_\Omega^{Cl}(\hat{x})$. Since $f^\circ(\hat{x}; v) \geq 0$ for all $v \in T_\Omega^{Cl}(\hat{x})$, it follows that $(-\nabla f(\hat{x}))^T v \leq 0$, and the second result follows from Definition 3.3.5. Furthermore,

if Ω is regular at $\hat{x}$, then by Definition 3.3.4, $T_\Omega^{Cl}(\hat{x}) = T_\Omega^{Co}(\hat{x})$, and the final two results follow directly from Definition 3.3.5. $\square$

The next result is similar to Theorem 3.3.8 but considers the limit of EXTENDED POLL endpoints $\hat{z}$ instead of $\hat{x}$.

Theorem 3.3.9. *Assume that $\hat{y} \in \mathcal{N}(\hat{x})$, $T_\Omega^H(\hat{z}) \neq \emptyset$, and the set of refining directions is asymptotically dense in $T_\Omega^H(\hat{z})$.*

1. *If f is Lipschitz near $\hat{z}$ with respect to the continuous variables, then $\hat{z}$ is a Clarke stationary point of f on Ω with respect to the continuous variables.*
2. *If f is strictly differentiable at $\hat{z}$ with respect to the continuous variables, then $\hat{z}$ is a Clarke KKT stationary point of f on Ω with respect to the continuous variables.*

Furthermore, if Ω is regular at $\hat{z}$, then the following hold:

1. *If f is Lipschitz near $\hat{z}$ with respect to the continuous variables, then $\hat{z}$ is a contingent stationary point of f on Ω with respect to the continuous variables.*
2. *If f is strictly differentiable at $\hat{z}$ with respect to the continuous variables, then $\hat{z}$ is a contingent KKT stationary point of f on Ω.*

Proof. The proof is identical to that of Theorem 3.3.8, but with $\hat{z}$ replacing $\hat{x}$. $\square$

3.3.2 Convergence of MV-MADS with Ranking and Selection (MVMADS-RS). Convergence for mixed variable MADS for the stochastic case has not yet been formally proven. Because of the similarities between MV-MADS and MGPS-RS discussed in Section 3.1.3 and the rigorous convergence results that exist for MGPS-RS, the following conjecture is made with the belief that a proof similar to that for MGPS-RS would follow for MVMADS-RS.

Conjecture 3.3.10. *Suppose the sequence of iterates generated by MVMADS-RS converges to $\hat{x} \in \Omega$. Then $\hat{x}$ meets the first-order necessary conditions (in the forms listed below) for optimality a.s.:*

- *If f is Lipschitz near $\hat{x}$, then $\hat{x}$ is a Clarke stationary point of f on Ω.*
- *If f is strictly differentiable at $\hat{x}$ and $T_\Omega^H(\hat{x}) \neq \emptyset$, then $\hat{x}$ is a Clarke KKT stationary point of f over Ω.*
- *If f is strictly differentiable at $\hat{x}$, Ω is regular at $\hat{x}$, and $T_\Omega^H(\hat{x}) \neq \emptyset$, then $\hat{x}$ is a contingent KKT stationary point of f over Ω.*

3.3.3 Convergence Results for Multi-Objective Problem. Convergence results for the multi-objective problem are now presented. First, it is shown that if the subproblems converge to an optimal solution, that solution is also Pareto optimal.

Lemma 3.3.11. *Given a feasible point $x^* \in \Omega$ of the stochastic, multi-objective, mixed-variable optimization problem defined in Equation (1.5), if x^* is a global minimizer of a convex combination of the J objectives, then x^* is Pareto optimal.*

Proof. Assume to the contrary that x^* is not Pareto optimal. If x^* is not Pareto optimal, by *Definition 1.1.3*, there exists some $x \in \Omega$ such that $F_k(x) \leq F_k(x^*)$ for $k = 1, 2, \ldots, J$ and $F_i(x) < F_i(x^*)$ for some $i \in \{1, 2, \ldots, J\}$. Thus, the positive sum, $\sum_{i=1}^{J} c_i F_i(x) < \sum_{i=1}^{J} c_i F_i(x^*)$, which contradicts the assumption that $x^* = \arg\min_{x \in \Omega} \left(\sum_{i=1}^{J} c_i F_i(x) \right)$. Therefore, x^* is Pareto optimal. □

3.3.3.1 Linearly Constrained Problems. Convergence results for SMO-MADS depend entirely on the results for the sub-problems. The following assumptions are made [139]:

1. All iterates X_k produced by the the MGPS-RS algorithm lie in a compact set.
2. The objective function f is continuously differentiable with respect to the continuous variables when the discrete variables are fixed.
3. For each set of discrete variables X^d, the corresponding set of directions $D^i = G_i Z_i$ includes tangent cone generators for every point in Ω^c.
4. The rule for selecting directions D_k^i conforms to Ω^c for some $\varepsilon > 0$.

5. For each $q = 1, 2, \ldots, n_C$, the responses $\{F_{qs}\}_{s=1}^{s_q}$ are independent, identically and normally distributed random variables with mean $f(X_q)$ and unknown variance $\sigma_q^2 < \infty$, where $\sigma_l^2 \neq \sigma_q^2$ whenever $l \neq q$.

6. For the r-th R&S procedure with candidate set $C = \{Y_1, Y_2, \ldots, Y_{n_C}\}$, $RS(C, \alpha_r, \delta_r)$ guarantees correctly selecting the best candidate $Y_{[1]} \in C$ with probability of at least $1 - \alpha_r$ whenever $f\left(Y_{[q]}\right) - f\left(Y_{[1]}\right) \geq \delta_r$ for any $q \in \{2, 3, \ldots, n_C\}$.

7. For all but a finite number of MGPS-RS iterations and sub-iterations, the best solution $Y_{[1]} \in C$ is unique, *i.e.*, $f(Y[1]) \neq f\left(Y_{[q]}\right)$ for all $q \in \{2, 3, \ldots, n_C\}$ where $C = \{Y_1, Y_2, \ldots, Y_{n_C}\} \subset M_k$ at iteration k.

Given a feasible point $x^* \in \Omega$ of a stochastic, multi-objective, mixed-variable optimization problem as defined in Equation (1.5), where the continuous variables in Ω are restricted by bound and linear constraints, the following key result applies.

Theorem 3.3.12. *The sequence of iterates generated by each subproblem of stochastic multi-objective pattern search (SMOMADS) contains a limit point that meets the first-order necessary conditions for Pareto optimality, almost surely (a.s.).*

Proof. The SMOMADS algorithm generates each subproblem as a nonnegative combination of the J objectives of the original problem, *i.e.*, $Z(x) = \left(\sum_{i=1}^{J} c_i F_i(x)\right)$, $c_i \geq 0$. Each subproblem is then solved using MGPS-RS. Results from Theorem 4.8 and Theorem 4.13 [139] show that the sequence of iterates produced in the subproblem contains a limit point x^* satisfying first-order conditions for optimality *a.s.* By Lemma 3.3.11, if x^* is optimal, it is also Pareto optimal. □

3.3.3.2 Nonlinearly Constrained Problems. The following assumptions are made [5]:

1. All iterates X_k produced by the the MVMADS-RS algorithm lie in a compact set.

2. The objective function f is continuously differentiable with respect to the continuous variables when the discrete variables are fixed.

3. For each $q = 1, 2, \ldots, n_C$, the responses $\{F_{qs}\}_{s=1}^{s_q}$ are independent, identically and normally distributed random variables with mean $f(X_q)$ and unknown variance $\sigma_q^2 < \infty$, where $\sigma_l^2 \neq \sigma_q^2$ whenever $l \neq q$.

4. For the r-th R&S procedure with candidate set $C = \{Y_1, Y_2, \ldots, Y_{n_C}\}$, $RS(C, \alpha_r, \delta_r)$ guarantees correctly selecting the best candidate $Y_{[1]} \in C$ with probability of at least $1 - \alpha_r$ whenever $f\left(Y_{[q]}\right) - f\left(Y_{[1]}\right) \geq \delta_r$ for any $q \in \{2, 3, \ldots, n_C\}$.

5. For all but a finite number of MVMADS-RS iterations and sub-iterations, the best solution $Y_{[1]} \in C$ is unique, *i.e.*, $f(Y[1]) \neq f\left(Y_{[q]}\right)$ for all $q \in \{2, 3, \ldots, n_C\}$ where $C = \{Y_1, Y_2, \ldots, Y_{n_C}\} \subset M_k$ at iteration k.

Theorem 3.3.13. *Suppose the sequence of iterates generated by a subproblem of SMOMADS solved using MVMADS-RS converges to $\hat{x} \in \Omega$. Then $\hat{x}$ meets the first-order necessary conditions (in the forms listed) for optimality a.s.:*

- *If f is Lipschitz near $\hat{x}$, then $\hat{x}$ is a Clarke stationary point of f on Ω*
- *If f is strictly differentiable at $\hat{x}$ and $T_\Omega^H(\hat{x}) \neq \emptyset$, then $\hat{x}$ is a Clarke KKT stationary point of f over Ω.*
- *If f is strictly differentiable at $\hat{x}$, Ω is regular at $\hat{x}$, and $T_\Omega^H(\hat{x}) \neq \emptyset$, then $\hat{x}$ is a contingent KKT stationary point of f over Ω.*

Further, if $\hat{x}$ is in fact optimal, it is also Pareto optimal.

Proof. The SMOMADS algorithm generates each subproblem as a nonnegative combination of the J objectives of the original problem, *i.e.*, $Z(x) = \left(\sum_{i=1}^{J} c_i F_i(x)\right)$, $c_i \geq 0$. Each subproblem is then solved using MADS. Thus, by Conjecture 3.3.10, the limit point $\hat{x}$ satisfies first-order necessary conditions for optimality, *i.e.*, is a stationary point, *a.s.* Therefore, by Lemma 3.3.11, if $\hat{x}$ is optimal, it is also Pareto optimal. □

3.4 Summary

In this chapter, the elements of the SMOMADS algorithm—R&S, MGPS-RS, MADS, and aspiration/reservation level analysis—have been described and the algo-

rithm itself has been outlined and described. Additionally, it has been shown that each subproblem of the SMOMADS algorithm converges almost surely to stationary points appropriately defined in the mixed variable domain, and that if such points are globally optimal, then they are also Pareto optimal. After the methodology was developed and theoretical convergence properties determined, algorithmic implementations were developed for testing. These implementations are given in Chapter IV.

IV. Algorithmic Implementation

In this chapter, the details of the various SMOMADS algorithm implementations are presented. The implementations can be categorized based on four central issues: determining aspiration/reservation level pairings in the objective space (*i.e.*, in the master problem), type of stochastic solver, type of solution filter, and termination criteria. These issues are discussed in Sections 4.1–4.4, respectively. The implementations are described in Section 4.5.

4.1 Objective Space Search Pattern

As discussed in Section 3.2.1.1, specific choices of aspiration and reservation levels are used to combine the multiple objectives of the original problem into a single objective subproblem, which generates a point on the approximate Pareto front in a given region of interest. In generating an approximate Pareto front, it is important that efficient points not be too clustered in the same region, so that the true Pareto front can be more fully and accurately characterized. Principles of experimental design were used in the objective space to accomplish this goal. Specifically, three methods were chosen, and in each case, the levels of the design were chosen to be specific predetermined values for the aspiration and reservation levels.

1. **Full Factorial Design.** The full factorial design (FFD) has as design points every possible combination of preselected design variable values. Though full factorial designs provide information about linear, interaction, and quadratic effects (*i.e.*, in characterizing the shape of the Pareto front), they become impractically large for relatively few numbers of design variables and levels because the number of design variables grows twice as fast as the number of objective functions.

2. **Central Composite Design.** The central composite design (CCD) is a five-level variance-optimal design used to fit second order models. It is considered by proponents of experimental design [104] to be quite useful for sequential experimentation, in which lower cost screening experiments are first performed

to determine regions where further scrutiny is required [104]. Information about shape of the Pareto front can be determined with relatively few design points [104].

3. **Box-Behnken Design.** The Box-Behnken design (BBD) was developed as a three-level alternative to the CCD. It is a spherical design that provides good coverage of the design space in general. However, because it is spherical, instead of cuboidal, it should not be used if the decision maker is particularly concerned with the extreme points of the given domain [25, 104].

Though only these three methods were implemented in this research, other experimental designs are available and may be of use. Additional insights about the master problem (*i.e.*, the Pareto front) may allow the user to more parsimoniously allocate potentially costly objective function evaluations (or simulations). Further research in this area is suggested in Section 6.2.3.

4.2 Stochastic Solvers

Once the objectives of the master problem are combined into a single objective, the MGPS-RS and MVMADS-RS stochastic optimization algorithms, discussed in Sections 3.1.2 and 3.2.1.4, respectively, were applied because of the their attractive convergence properties and because they are useful on industrial problems in which derivatives are not available (*e.g.* see [144], [17] or [4]). The actual implementation makes use of the NOMADm [3] MATLAB® software, which has been applied successfully to several single-objective stochastic test problems [45] and a multi-echelon repair system optimization application [144].

NOMADm was extended to include the multi-objective case in the following way. Code for interactive specification of aspiration/reservation level analysis was required to control the master problem. Since no suitable MATLAB code exists for doing this, new code, called *MOMADS* (see Appendix A), was written, which incorporates the NOMADm software as the single-objective stochastic subproblem

solver as described in Section 3.2. A copy of this new code is included as Appendix A.

4.3 Solution Filter

For deterministic optimization problems, the solutions to the subproblems of the SMOMADS algorithm are guaranteed to be efficient, or non-dominated, points (see Section 3.3.3). This is also true of stochastic optimization problems, but only in the limit, *i.e.*, if the subproblem is allowed an infinite number of iterations. However, since an infinite number of iterations is not possible in practice, some non-efficient solutions may enter the efficient set. For this reason, each point is checked for Pareto dominance before entering the efficient set, and if it is dominated, it is "filtered" and does not enter the efficient set.

4.4 Algorithm Termination Criteria

In SMOMADS, algorithmic termination must occur both in the subproblem and in the master problem. Termination criteria for the subproblems is discussed in Section 4.4.1 and termination criteria for the master problem is discussed in Section 4.4.2.

4.4.1 Terminating the Subproblems. Even if an algorithm is known to converge, the reality of imprecision and roundoff error make it necessary to predetermine stopping criteria. In pattern search methods, this is accomplished by stopping the algorithm when the mesh size Δ_k^m falls below a threshold value Δ_T; *i.e.*, $\Delta_k \leq \Delta_T$ [67,137]. This criterion has been used extensively in practice, and Dolan *et al.* [44] showed that for problems without noise Δ_k^m provides a measure of first-order stationarity; *i.e.*, $\|\nabla f(x_k)\| \leq C\Delta_k^m$, where $C > 0$ is an unknown constant. Similar termination criteria may be used for MADS.

In a stochastic environment, termination criteria are typically more complex. Too small a value of Δ_T may increase the required number of function evaluations

required by the ranking and selection portion of the algorithm to an unacceptable level, whereas, too large a value may induce premature termination. For MGPS-RS applied to single objective problems two additional termination criteria are employed [137]. The first is that the ratio between the response standard deviation S and the indifference zone parameter δ_r exceeds some threshold; namely,

$$\frac{S}{\delta_r} \geq \sqrt{K}, \tag{4.1}$$

where K may be selected by the user, based on available sampling budget; larger values of K allow larger budgeting thresholds [137]. This criterion is a measure of signal-to-noise. It is a heuristic for signalling that a high response noise (compared to the indifference zone parameter) is causing excessively large sampling requirements [137]. The second criterion is that the significance level α_r in the ranking and selection procedure satisfy

$$\alpha_r \leq \alpha_T,$$

where α_T is a threshold of error to ensure that the probability of correct selection $1 - \alpha_T$ from among the candidates is sufficiently high. This is intended to prevent a sequence of erroneous selections from causing the step size to decrease to a small value thereby prematurely terminating the algorithm.

4.4.2 Terminating the Master Problem. In Section 4.1 experimental designs were developed to search the objective space. In those implementations, a fixed number of design points are determined *a priori* and the algorithm terminates when these experimental design points are exhausted; however, in some cases this may not be the ideal method for terminating the master problem. For example, in the full factorial design, the number of design points can be prohibitively large. If a metric to determine the quality of the Pareto set existed, it could be used to terminate the algorithm without running all the design points if the Pareto set was determined to be "adequate". Therefore, one area of future work is to apply the quality metrics

introduced by Wu and Azarm [163] to assess the Pareto set.

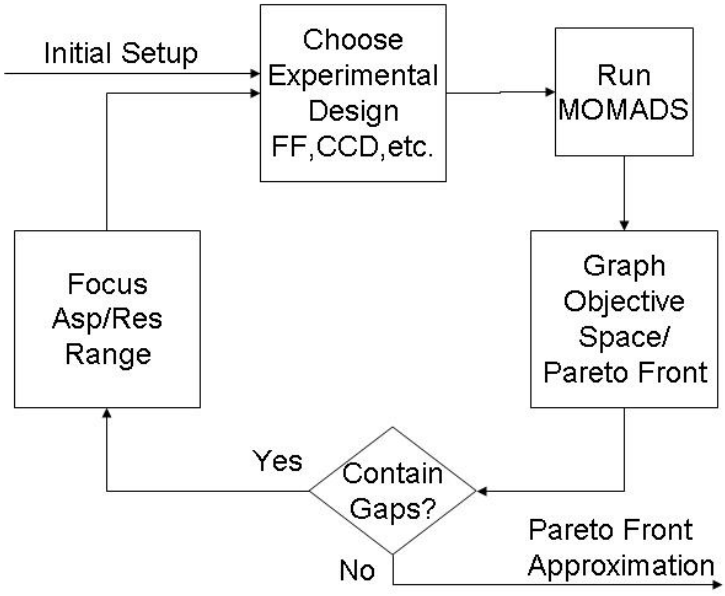


Figure 4.1: Master problem algorithm with termination check (graphical)

Termination in the master problem was accomplished via the algorithm depicted in Figure 4.1 and described in Figure 4.2. The approach is not ideal, as it requires intuitive observation on the part of an experienced analyst with moderate insight into the expected objective space; however, for the purpose of this initial study, it does perform quite well. Additionally, in many real-world engineering problems, extreme cases are not typically considered; *e.g.*, an aircraft designed to have zero reliability or excessively high cost. Furthermore, some information about the decision maker's region of interest is likely to exist and provide insight into whether to expend (objective function evaluation) resources to fill gaps in regions of little or no interest to the decision maker. So, though not theoretically ideal, this approach is usually adequate in practice.

It has been suggested previously that the quality metrics, applied *a posteriori* in this study, could also be applied after each successive design point. This would provide insight into the quality of the Pareto set approximation and quantitative termination criteria. However, this would only provide information about when to

A Method For Determining When to Terminate the Master Problem

1. Choose initial SMOMADS implementation based on problem structure (types of constraints, types and number of objectives, etc.).
2. Run the implementation chosen in step 1 and graph the results in the objective space. (For problems with more than three objectives, the objectives can be graphed pairwise.)
3. Visually inspect the approximated Pareto front and consider the following:
   - Do gaps appear?
   - Is the spread adequate for the desired range of aspiration and reservation levels?
   - Are desired "extreme solutions " represented, *i.e.*, those solutions close in one objective to its component in the utopia point?
   - Are solutions clustered?
4. If approximation is adequate, then keep current efficient set and run quality metrics. Retain current Pareto set and stop.
5. Otherwise, determine region requiring further scrutiny, *e.g.*, regions between points **A1** and **R1** and between points **A2** and **R2** in Figure 4.3.

Figure 4.2: Master problem algorithm with termination check

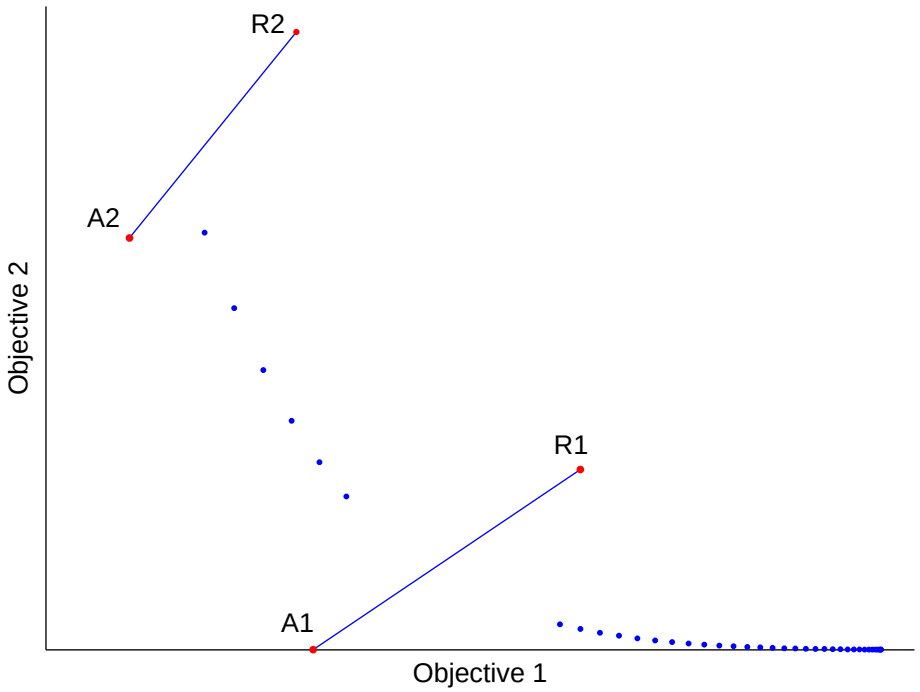


Figure 4.3: Examining missing regions of the Pareto front

terminate. If the set is determined to be inadequate, the metrics by themselves would not indicate how to set the aspiration and reservation levels to improve the approximation in successive runs. The analyst would still be needed to determine appropriate settings.

4.5 SMOMADS Implementations

Seven instances of the SMOMADS algorithm, defined by the choices discussed in Sections 4.1–4.4 were devised and tested. They are listed in Table 4.1, where columns 2-4 describe the choice that define the number listed in column 1.

4.6 Testing the Algorithm

In this chapter, implementation of the SMOMADS algorithm has been described in detail. In particular, four central issues were addressed: search pattern in the

Table 4.1: SMOMADS Implementations Tested

Number	Solver	Experimental Design	Filter	Test Problems	Section
1	MGPS-RS	Full Factorial	No	Viennet 4	*5.1.1.1*
2	MGPS-RS	Full Factorial	Yes	Viennet 3	*5.1.1.2*
				Poloni	*5.1.1.3*
				Dias Γ1	*5.1.2.1*
				Dias Γ2	*5.1.2.2*
				Fonseca F1	*5.1.2.3*
				Schaffer F3	*5.1.2.4*
				DTLZ7	*5.1.2.6*
3	MGPS-RS	Central Composite	No	Viennet 4	*5.1.1.1*
4	MGPS-RS	Central Composite	Yes	Dias Γ1	*5.1.2.1*
				Fonseca F1	*5.1.2.3*
				Schaffer F3	*5.1.2.4*
				DTLZ7	*5.1.2.6*
5	MGPS-RS	Box-Behnken	No	Viennet 4	*5.1.1.1*
6	MVMADS-RS	Full Factorial	Yes	Tamaki	*5.1.1.4*
				Srinivas	*5.1.2.5*
7	MVMADS-RS	Central Composite	Yes	Tamaki	*5.1.1.4*
				Srinivas	*5.1.2.5*
				Disk Brake	*5.1.3.1*

objective space (*i.e.*, in the master problem), type of stochastic solver, termination criteria, and whether or not a solution filter (as described in Section 3.2.1.5) was used. The seven implementations shown in Table 4.1 were used to solve published multi-objective test problems with known results and then applied to a multi-objective design optimization problem. The test problems, application problem, and results are given in Chapter V.

V. Computation and Evaluation

The accuracy of the SMOMADS algorithm was verified by its use on a set of test problems with known solutions. These test problems were modified (via the MATLAB® `rand` function) to introduce random noise into each objective function to simulate the algorithm's use on a stochastic system. The problems (except the airfoil design problem) were solved in Matlab 7.2.0 on a 2.13 GHz Pentium(R)M processor with 1GB of RAM. Initially, a prototype of the MOMADS code (see Appendix A) was verified by its use on one test problem [149] before checking an entire set of problems. Results of the initial test, which are also included in [150], are shown in Section 5.1.1.1.

Following positive initial results, additional test problems were identified targeting several characteristics: bi-objective, multi-objective, continuous-variable, mixed-variable, discontinuous Pareto front. Specific test problems are listed in Table 5.1. Results for each test problem are shown in Section 5.1 and associated quality metrics are listed in Table 5.3 in Section 5.3.

Table 5.1: Published Problems Tested using SMOMADS

Test Problem	# Var	# Obj	Type of Variables	Sequential Experiment	# Test Points	Results in Section
Viennet 4	2	3	Continuous	No	4,209	*5.1.1.1*
Viennet 3	2	3	Continuous	No	4,096	*5.1.1.2*
Poloni	2	2	Continuous	Yes	10,272	*5.1.1.3*
Tamaki	3	3	Continuous	Yes	145	*5.1.1.4*
Dias Γ1	30	2	Continuous	Yes	697	*5.1.2.1*
Dias Γ2	30	2	Continuous	No	625	*5.1.2.2*
Fonseca F1	2	2	Continuous	Yes	10,036	*5.1.2.3*
Schaffer F3	1	2	Continuous	Yes	11,250	*5.1.2.4*
Srinivas	2	2	Continuous	Yes	697	*5.1.2.5*
DTLZ7	2	2	Continuous	No	36	*5.1.2.6*
Disk Brake	4	2	Mixed	Yes	108	*5.1.3.1*

The algorithm was then applied to an airfoil design optimization problem. This problem is representative of real-world multi-objective aircraft design problems. (Also see [35], [77], [90], and [82].) Additionally, objectives of engineering design optimization problems are often either subject to measurement error or must be estimated

with simulations. Examples of simulation used in aircraft design can be found in [91], [81], [36], and [35]. Results of the application are shown in Section 5.2.

5.1 Test Results

5.1.1 Continuous Multi-Objective Test Problems.

5.1.1.1 Viennet 4. As a proof-of-concept, Implementation 1 of the algorithm (see Section 4.5) was built and tested on a problem given by Viennet and Marc [149]:

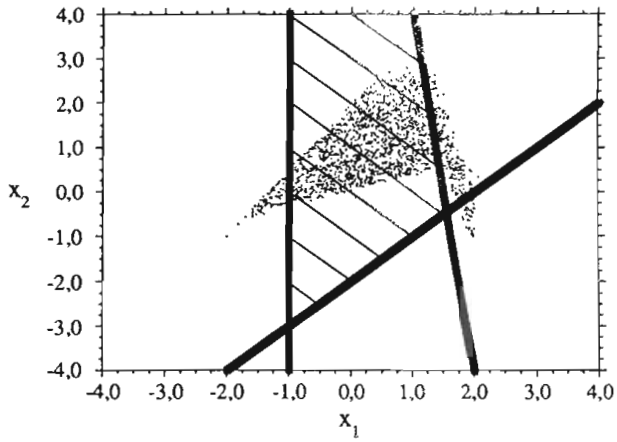
$$\begin{aligned}
\min\ F_1(X_1, X_2) &= \frac{(X_1 - 2)^2}{2} + \frac{(X_2 + 1)^2}{13} + 3 \\
F_2(X_1, X_2) &= \frac{(X_1 + X_2 - 3)^2}{175} + \frac{(2X_2 - X_1)^2}{17} - 13 \\
F_3(X_1, X_2) &= \frac{(3X_1 - 2X_2 + 4)^2}{8} + \frac{(X_1 - X_2 + 1)^2}{27} + 15
\end{aligned}$$

subject to

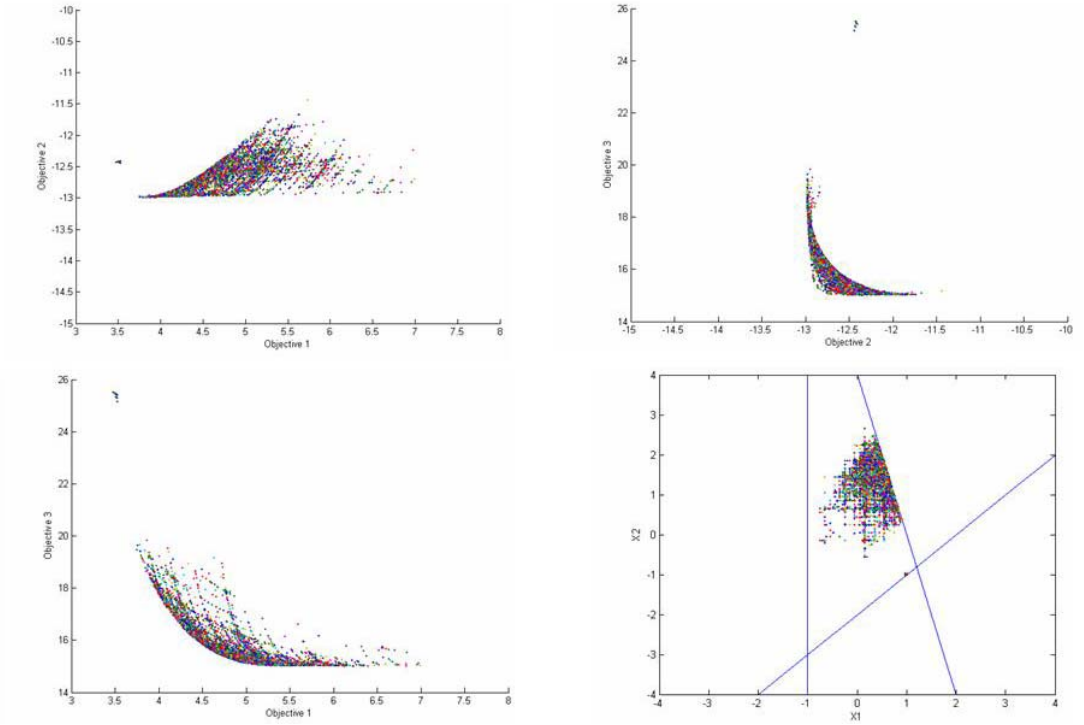
$$\begin{aligned}
4X_1 + X_2 - 4 &\leq 0 \\
-X_1 - 1 &\leq 0 \\
X_1 - X_2 - 2 &\leq 0 \\
X_1, X_2 &\in [-4, +4]^2.
\end{aligned}$$

This test problem was modified by adding uniformly distributed random noise between 0 and 1 to each objective function evaluation to simulate the algorithm's use on a stochastic system[1]. The algorithm was tested on the initial problem over a range of aspiration and reservation levels using three different experimental designs: CCD with 59 design points, BBD with 54 design points, and FFD with 3 objectives set at 4 levels within the region of interest, resulting in 4,096 design points. Five replications were used at each design point. Each run took less than a minute (with 500 function evaluations).

[1]This is true of all further test problems as well and thus will not be mentioned in each further test problem.



(a) Pareto Set for Deterministic Test Problem, Figure 7 in [149]



(b) Initial Test Results for Test Problem with Added Noise using Full Factorial Design

Figure 5.1: Comparison of Initial Test Results to Published Solution

Initial results are shown in Figure 5.1. The initial runs fall inside the published Pareto set (see Figure 5.1(a)), implying that Stage 1 of the algorithm is indeed converging to Pareto solutions. Because this problem contains 3 objective functions, pairwise examinations of the experimental Pareto fronts are used. These appear to approach a reasonable approximation to the actual pairwise Pareto fronts. However, the effect of random noise appears to be more prominent in objective 2. This is expected because the random noise was not scaled and is much larger in comparison to values of objective 2 than compared to those of objectives 1 and 3. In the remaining test problems, noise for each objective was scaled to approximately one percent of its maximum objective function value (*i.e.*, value at the nadir point) to account for differences in the relative sizes of the objective function values, so that each is affected similarly by the noise.

In general, as the magnitude of the noise grows, the number of iterations required before the algorithm terminates may become prohibitively large. As discussed in Section 4.4.1, one of the stopping criteria in the subproblems is the signal-to-noise ratio squared (K in Equation (4.1)). Assuming a computational budget as a function of this value K, the sample variance should be less than $K\delta_r^2$, where δ_r^2 is the desired indifference zone parameter. Additionally, because the number of iterations is necessarily finite, the filter as described in Section 3.2.1.5 was added to all further implementations (Implementations 2, 4, 6, and 7 in Table 4.1) to prevent potentially dominated points from entering the efficient set.

5.1.1.2 Viennet 3 Test Problem. The Viennet 3 test problem [149] is linearly constrained and multi-objective and is given by

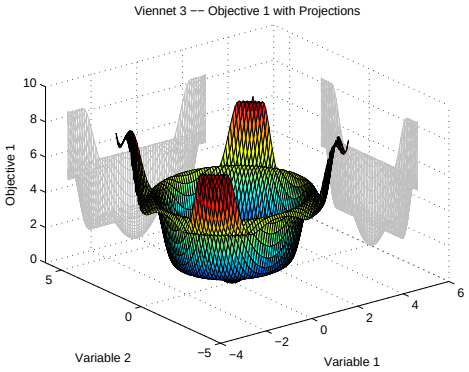
$$\begin{aligned} \min \; F_1(x,y) &= 0.5\left(x^2+y^2\right) + \sin\left(x^2+y^2\right) \\ F_2(x,y) &= \frac{\left(3x-2y+4\right)^2}{8} + \frac{\left(x-y+1\right)^2}{27} + 15 \\ F_3(x,y) &= \frac{1}{\left(x^2+y^2+1\right)} - 1.1e^{\left(-x^2-y^2\right)} \end{aligned}$$

subject to

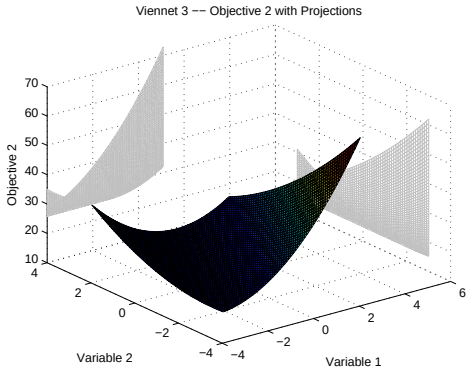
$$-3 \leq x, y \leq 3.$$

Graphical depictions of the objective functions can be seen in Figure 5.2. Using implementation 2 (see Section 4.5), the SMOMADS algorithm found an approximation to the Pareto optimal set and front, shown in Figures 5.3(c) and 5.3(d) respectively, by varying the aspiration and reservation levels using a full factorial design with 3 objectives set at 4 levels, resulting in 4,096 design points. Because solutions returned by the subproblem may be filtered and not enter the Pareto set, five replications were used at each design point.

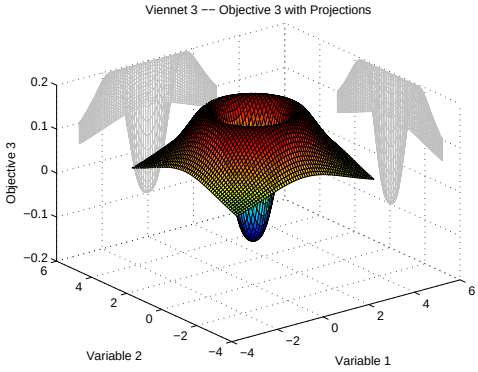
When compared to the published results from [148,149] shown in Figures 5.3(a) and 5.3(b), it is clear that the SMOMADS algorithm was able to produce a good approximation of the published solution. Although each subproblem replication was relatively fast, the large number of design points associated with the full factorial design led to several hours of computation. This time could be reduced by using a central composite design to examine the entire space and then focus in areas where gaps may exist in the approximation to the Pareto front. Additionally, test metrics, defined in Section 3.2.1.6, were used to determine the quality of the Pareto set after the run was complete. These values are shown in Table 5.3. However, because these metrics are generally used to compare Pareto sets, it would be useful in determining if the algorithm should terminate before the entire experimental design had been executed if a desired quality in the Pareto set approximation or threshold value had



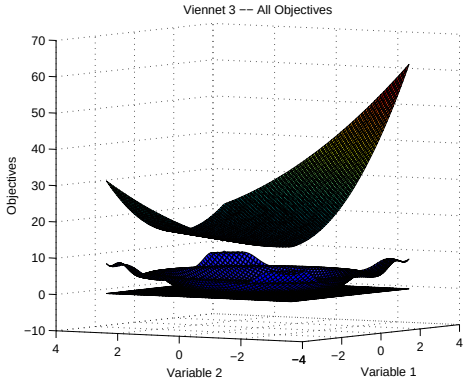
(a) Objective Function 1 with Projections onto the axes



(b) Objective Function 2 with Projections onto the axes

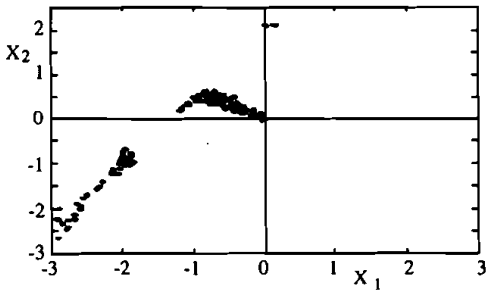


(c) Objective Function 3 with Projections onto the axes

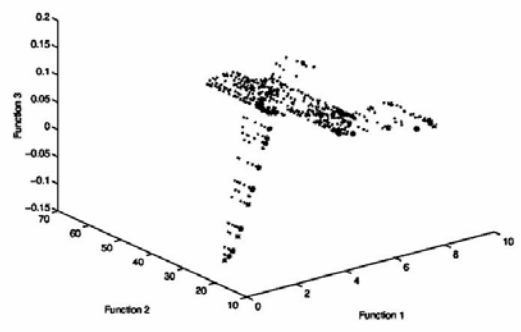


(d) All Objective Functions

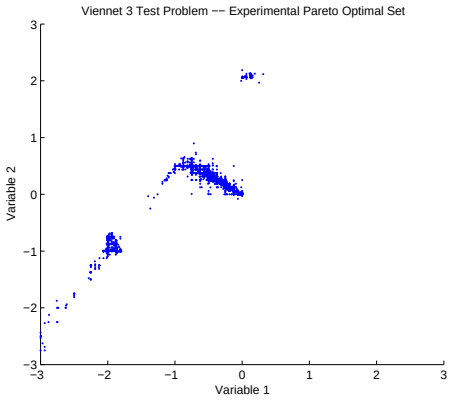
Figure 5.2: Graphical Depiction of the Viennet 3 Test Problem



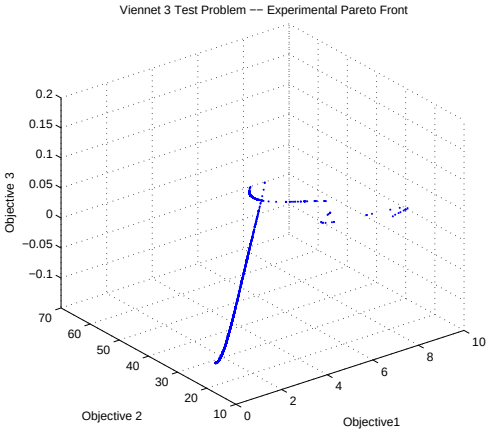
(a) Pareto Optimal Set for Deterministic Test Problem, Figure 6 in [149]



(b) Pareto Front for Deterministic Test Problem, Figure C.36 in [148]



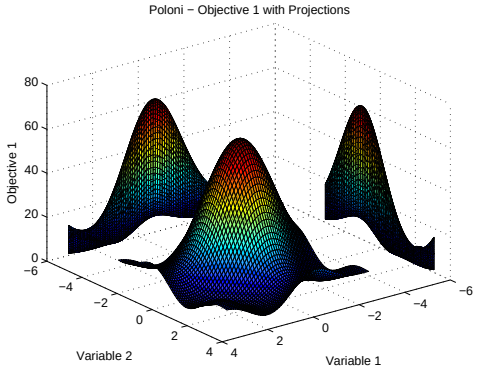
(c) Experimental Pareto Optimal Set



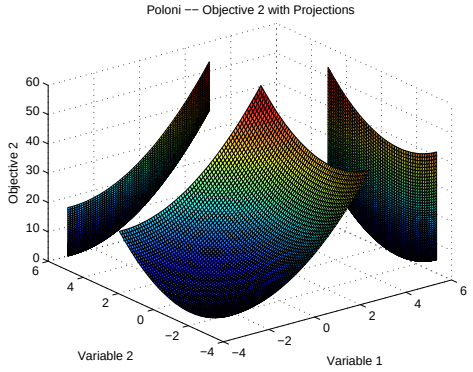
(d) Experimental Pareto Front

Figure 5.3: Comparison of Experimental Results to Published Solution for Viennet 3 Test Problem

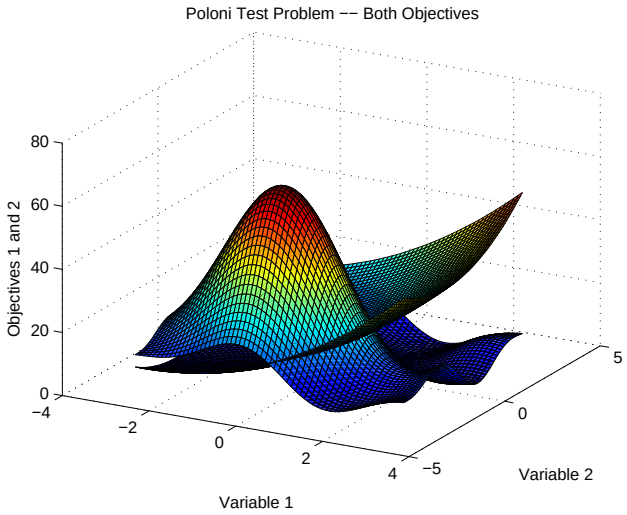
been obtained (see Section 4.4.2). Future research in this area is recommended in Section 6.2.3.2.



(a) Objective Function 1 with Projections onto the axes



(b) Objective Function 2 with Projections onto the axes



(c) Both Objective Functions

Figure 5.4: Graphical Depiction of the Poloni Test Problem

5.1.1.3 Poloni Test Problem. The Poloni test problem, introduced by Carlo Poloni in 1995 [117], contains two nonlinear objectives and simple bounds on

the variables. It is given by

$$\min F(x,y) = (f_1(x,y), f_2(x,y))$$

subject to

$$-\pi \leq x, y \leq \pi,$$

where

$$\begin{aligned}
f_1(x,y) &= 1 + (A_1 - B_1)^2 + (A_2 - B_2)^2 \\
f_2(x,y) &= (x+3)^2 + (y+1)^2 \\
A_1 &= 0.5\sin(1) - 2\cos(1) + \sin(2) - 1.5\cos(2) \\
A_2 &= 1.5\sin(1) - \cos(1) + 2\sin(2) - 0.5\cos(2) \\
B_1 &= 0.5\sin(x) - 2\cos(x) + \sin(y) - 1.5\cos(y) \\
B_2 &= 1.5\sin(x) - \cos(x) + 2\sin(y) - 0.5\cos(y),
\end{aligned}$$

and graphical depictions of the objective functions are shown in Figure 5.4[2]. Implementation 2 (see Section 4.1) of the SMOMADS algorithm was used to find the approximations of the Pareto optimal set and Pareto front shown in Figures 5.5(c) and 5.5(d), respectively. As with the Viennet 3 test problem, the objective space was divided and a full factorial design used. In this case, however, Implementation 2 tended not to find the Pareto points that heavily favored one objective or the other. That is, solutions found by the algorithm tended to lie in the "compromise area" of the curve shown in Figure 5.5(d) (area near the middle of the Pareto front where individual objectives are neither extremely high nor extremely low). After adjusting the range of the aspiration and reservation levels, additional runs resulted in the generation of new points on the lower right side of the curve.

The solution set was then compared to the one published in the doctoral dissertation of Van Veldhuizen [148]. Van Veldhuizen's approximations of the Pareto

[2]Note that the objective functions used here as a minimization problem are the negatives of the of those used in the original maximization problem. To compare with published results, the results in Figures 5.5(c) and 5.5(d) are converted to those of the original maximization problem.

optimal set and Pareto front for the deterministic problem (*i.e.*, without noise added), which was generated by an evolutionary algorithm, are shown in Figures 5.5(a) and 5.5(b), respectively. As can be clearly seen from Figure 5.5(b), this Pareto set approximation contains dominated points. In contrast, the Pareto set approximation found via SMOMADS contains few (if any) dominated points as shown in Figure 5.5(d). Though the Van Veldhuizen set contains more points in the approximation and covers more of the objective space, it should be noted that the SMOMADS algorithm was not run exhaustively. Additional aspiration-reservation level pairs can be run in order to determine additional portions of the Pareto frontier. The limiting factor is the number of design points in the master problem which can be evaluated. Thus, as previously mentioned, efficient methods for searching the objective space can be of paramount importance.

5.1.1.4 Tamaki Test Problem. The Tamaki test problem [148] contains three linear objectives, one nonlinear constraint, and nonnegativity constraints on the variables. Its formulation is given by

$$\max F(x, y, z) = (f_1(x, y, z), f_2(x, y, z), f_3(x, y, z))$$

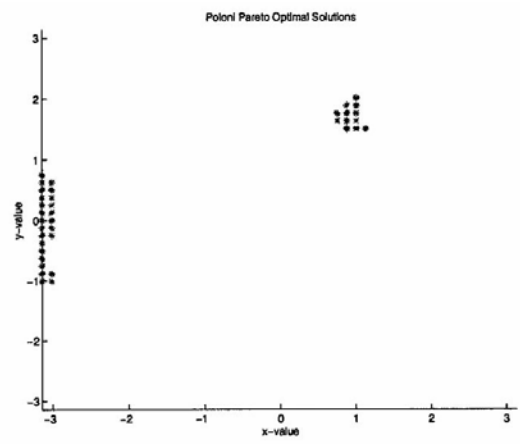
subject to

$$x^2 + y^2 + z^2 \leq 1$$
$$x, y, z \geq 0$$

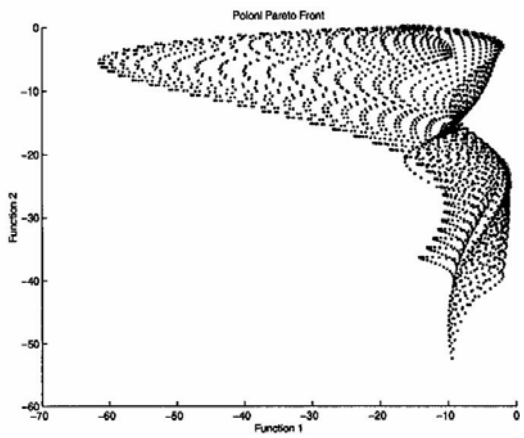
where

$$f_1(x, y, z) = x$$
$$f_2(x, y, z) = y$$
$$f_3(x, y, z) = z.$$

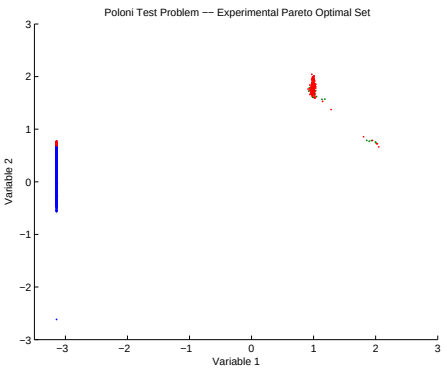
As shown in Figure 5.6(a), its feasible region is the nonnegative portion of a sphere. Because the objective is to maximize each variable, the true Pareto front is the surface of the sphere within the feasible region (see Figure 5.7(a)).



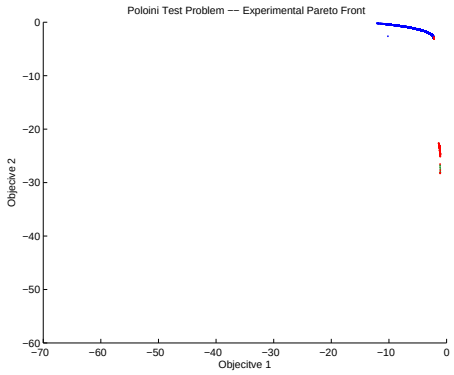
(a) Pareto Optimal Set for Deterministic Test Problem, Figure C.17 in [148]



(b) Pareto Front for Deterministic Test Problem, Figure C.18 in [148]



(c) Experimental Approximation of the Pareto Optimal Set, Determined Via Full Factorial Design of the Region of Interest



(d) Experimental Approximation of the Pareto Front, Determined Via Full Factorial Design of the Region of Interest Showing Few Dominated Points

Figure 5.5: Comparison of Experimental Results to Published Solution for Poloni Test Problem

Implementations 6 and 7 of the SMOMADS algorithm were used to find an approximation of the Pareto optimal set/front shown in Figures 5.7(c) and 5.7(d). As with previous test problems, the algorithm tended toward "compromise " regions of the objective space. However, this is due to the choice of experimental design in the master problem, which can be changed to use a different implementation of the algorithm, and is thus, not considered an algorithmic shortcoming.

5.1.2 Continuous Bi-Objective Test Problems.

5.1.2.1 Dias Γ1 *Test Problem.* The Dias Γ1 test problem is a bound constrained bi-objective (one linear function, one nonlinear function) problem with 30 decision variables. Its formulation is given by

$$\min F(X) = (f_1(\overrightarrow{X}), f_2(\overrightarrow{X}))$$

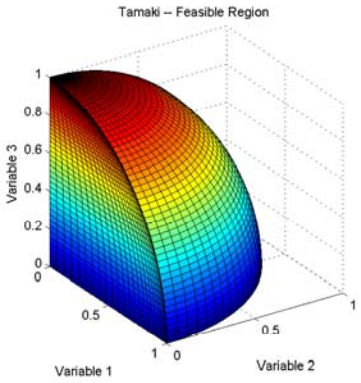
subject to

$$0 \leq X_i \leq 1, \quad i = 1, 2, \ldots, 30$$

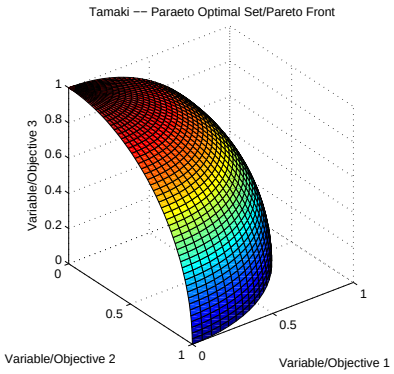
where

$$\begin{aligned} f_1(\overrightarrow{X}) &= X_1 \\ f_2(\overrightarrow{X}) &= \left[1 + 9\sum_{i=2}^{M}\left(\frac{X_i}{(M-1)}\right)\right]\left[1 - \sqrt{\frac{f_1(\overrightarrow{X})}{1 + 9\sum_{i=2}^{M}\left(\frac{X_i}{(M-1)}\right)}}\right] \\ \overrightarrow{X} &= [X_i, \ldots, X_{30}]. \end{aligned}$$

Implementation 2 of the SMOMADS algorithm was used to determine the approximation of the Pareto front (shown in Figure 5.8(b)), as follows. First, implementation 2 was run in the range between the utopia and nadir points in the objective space. The graph of the Pareto front was observed and areas with gaps and under-representation were determined. Further runs (using Implementation 4) were then focused on those areas by setting the aspiration and reservation range to a small area around the missing regions. When compared to the published solution to the deterministic version

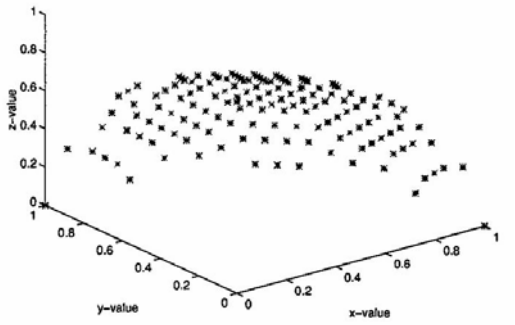


(a) Feasible Region

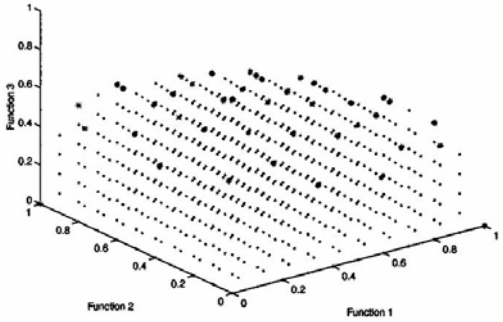


(b) Theoretical Pareto Front

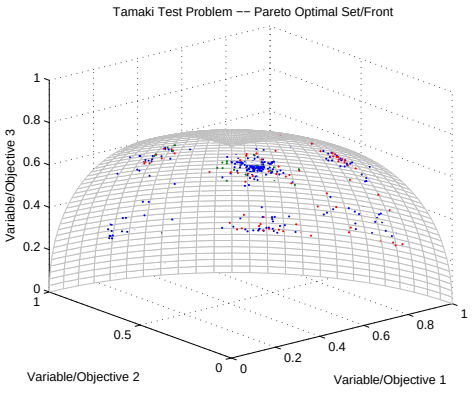
Figure 5.6: Theoretical Solutions for the Tamaki Test Problem



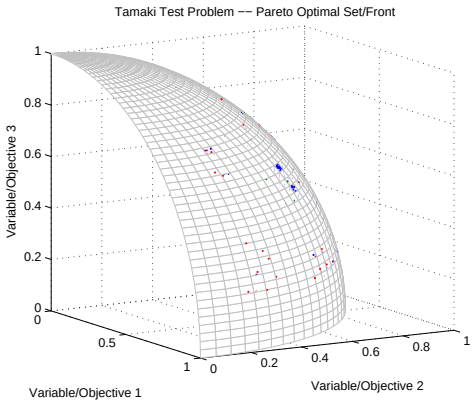
(a) Pareto Optimal Set for Deterministic Test Problem, Figure D.19 in [148]



(b) Pareto Front for Deterministic Test Problem, Figure D.20 in [148]



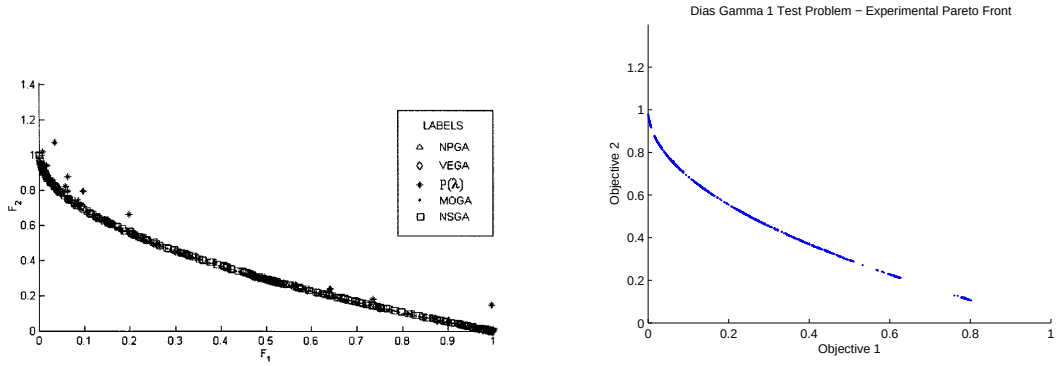
(c) Experimental Approximation of the Pareto Optimal Set and Front



(d) Experimental Approximation of the Pareto Set and Front

Figure 5.7: Comparison of Experimental Results to Published Solution for Tamaki Test Problem

of the problem found in [42], the approximation determined by SMOMADS is quite good, having nearly the same spread and fewer points that are actually dominated.



(a) Pareto Front for Deterministic Test Problem, Figure 2 in [42]

(b) Experimental Pareto Front

Figure 5.8: Comparison of Experimental Results to Published Solution for Dias' Γ1 Test Problem

5.1.2.2 Dias Γ2 Test Problem. The Diaz Γ2 problem is identical to the Diaz Γ1 problem, except for a change in the second objective. It is given by

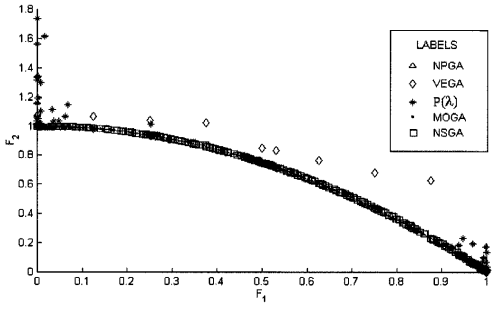
$$\min F(X) = (f_1(\overrightarrow{X}), f_2(\overrightarrow{X}))$$

subject to

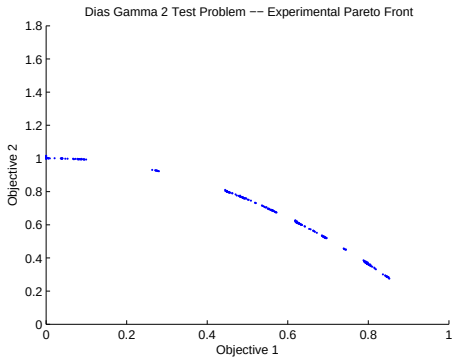
$$0 \leq X_i \leq 1, \quad i = 1, 2, \ldots, 30$$

where

$$\begin{aligned} f_1(\overrightarrow{X}) &= X_1 \\ f_2(\overrightarrow{X}) &= \left[1 + 9\sum_{i=2}^{M}\left(\frac{X_i}{(M-1)}\right)\right]\left[1 - \left(\frac{f_1(\overrightarrow{X})}{1 + 9\sum_{i=2}^{M}\left(\frac{X_i}{(M-1)}\right)}\right)^2\right] \\ \overrightarrow{X} &= [X_i, \ldots, X_{30}]. \end{aligned}$$



(a) Pareto Front for Deterministic Test Problem, Figure 3 in [42]



(b) Experimental Pareto Front

Figure 5.9: Comparison of Experimental Results to Published Solution for Dias' Γ2 Test Problem

5.1.2.3 Fonseca F1 Test Problem. The Fonseca F1 test problem contains two exponential objectives and two variables with upper and lower bounds. Graphs of both objective functions are shown in Figures 5.10(a) and 5.10(b), and the problem formulation is given by

$$\begin{aligned} \min \; f_1(X_1, X_2) &= 1 - \exp\left(-(X_1 - 1)^2 - (X_2 + 1)^2\right) \\ f_2(X_1, X_2) &= 1 - \exp\left(-(X_1 + 1)^2 - (X_2 - 1)^2\right) \end{aligned}$$

subject to

$$\begin{aligned} -2 \leq X_1 \leq 2 \\ -2 \leq X_2 \leq 2. \end{aligned}$$

The theoretical Pareto optimal set (as shown in Figure 5.11(a)) can be easily determined to be the line between points (0,1) and (1,0) [79]. The corresponding theoretical Pareto front is shown in Figure 5.11(b).

Implementations 2 and 4 of the SMOMADS algorithm were used to determine the experimental Pareto optimal set and front as shown in Figures 5.11(c) and 5.11(d), respectively. As shown in the graphs, the algorithm found a very good approximation in the "compromise region" as well as the extreme solutions. Gaps do exist in the

approximation that could be examined further via the master problem termination algorithm outlined in Section 4.4.2.

5.1.2.4 Schaffer F3 Test Problem. The Schaffer F3 test problem contains one quadratic objective function, one nonsmooth piecewise linear objective function, and one decision variable with simple bounds. The problem formulation is given by

$$\min F(X) = (f_1(X), f_2(X))$$

subject to

$$-8 \leq X \leq 8$$

where

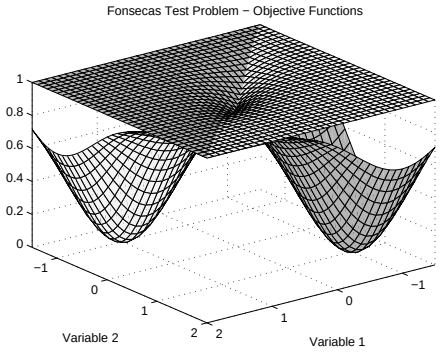
$$f_1(X) = \begin{cases} -X, & X \leq 1 \\ -2 + X, & 1 < X \leq 3 \\ 4 - X, & 3 < X \leq 4 \\ -4 + X, & 4 < X \end{cases}$$

$$f_2(X) = (X - 5)^2.$$

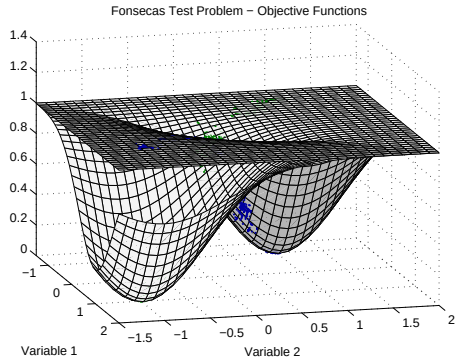
Graphical depictions of the objectives are shown in Figures 5.12(a) and 5.12(c). The published Pareto optimal set is also shown in Figure 5.12(a). The true Pareto front (derived from the published true Pareto set) is shown in Figure 5.12(b).

Implementations 2 and 4 were used to determine the experimental Pareto optimal set and front shown in Figures 5.12(c) and 5.12(d), respectively. As depicted in the graphs, the experimental solution nearly perfectly matches the published theoretical solution.

5.1.2.5 Srinivas Test Problem. The Srinivas test problem contains two nonlinear objective functions, two nonlinear constraints, and two decision variables

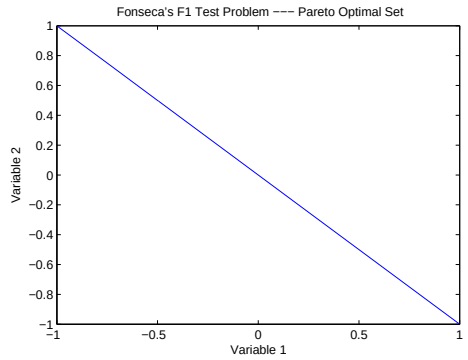


(a) Objective Functions

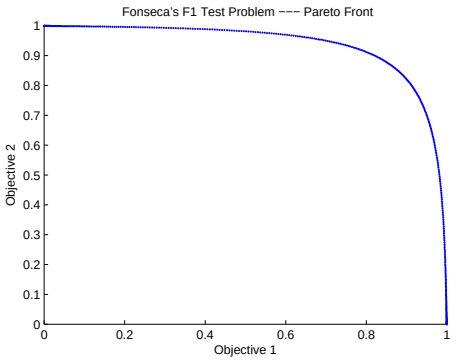


(b) Objective Functions – Alternate View

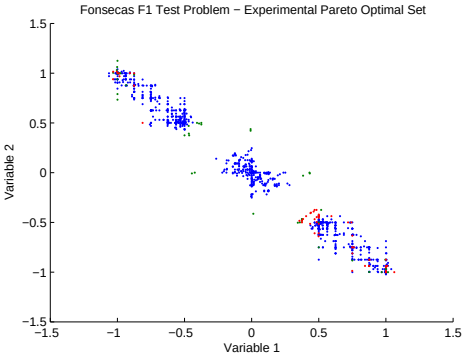
Figure 5.10: Graphical Depiction of the Fonseca F1 Test Problem



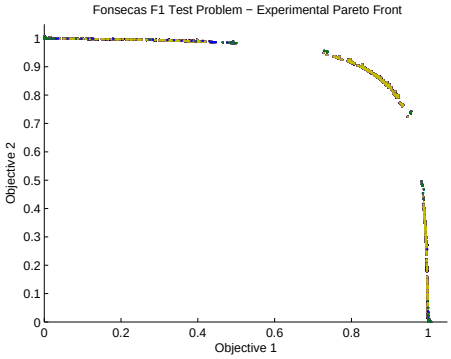
(a) Pareto Optimal Set for Deterministic Test Problem [79]



(b) Pareto Front for Deterministic Test Problem

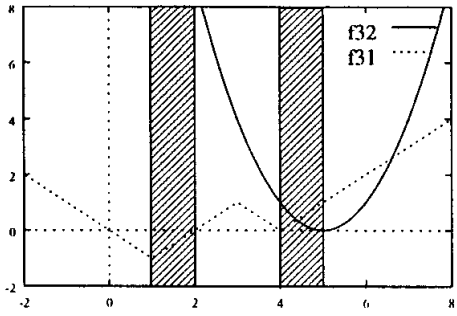


(c) Experimental Pareto Optimal Set

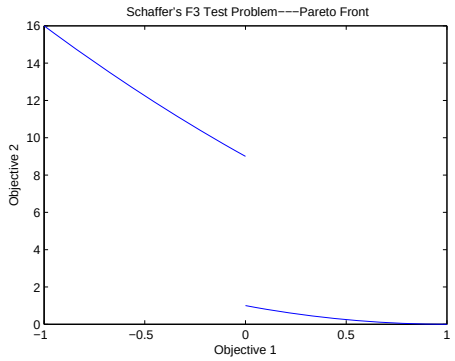


(d) Experimental Pareto Front

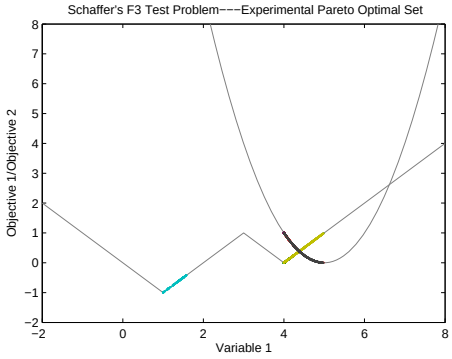
Figure 5.11: Comparison of Experimental Results to Published Solution for Fonseca F1 Test Problem



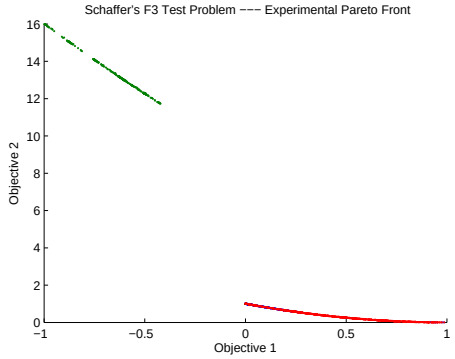
(a) Pareto Optimal Set for Deterministic Test Problem, Figure 4 in [79]



(b) Pareto Front for Deterministic Test Problem



(c) Experimental Pareto Optimal Set



(d) Experimental Pareto Front

Figure 5.12: Comparison of Experimental Results to Published Solution for Schaffer F3 Test Problem

with upper and lower bounds. Its formulation is given by

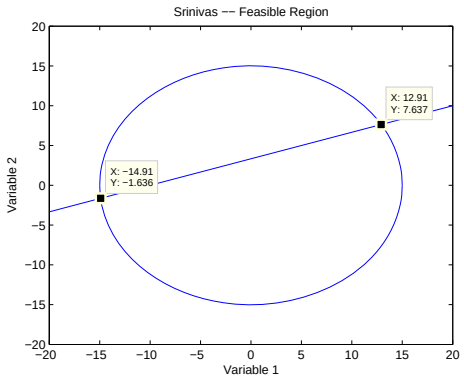
$$\begin{aligned} \min \quad & f_1(X_1, X_2) = 1 - \exp\left(-(X_1 - 1)^2 - (X_2 + 1)^2\right) \\ & f_2(X_1, X_2) = 1 - \exp\left(-(X_1 + 1)^2 - (X_2 - 1)^2\right) \end{aligned}$$

subject to

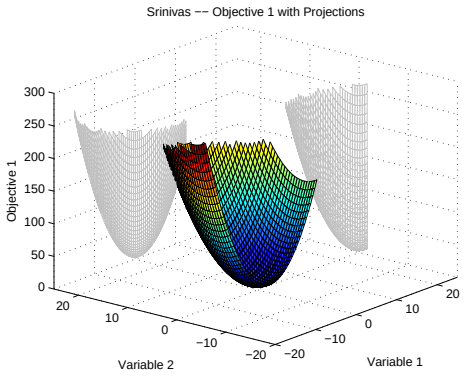
$$\begin{aligned} X_1^2 + X_2^2 - 225 &\leq 0 \\ X_1 - 3X_2 + 10 &\leq 0 \\ -20 \leq X_1 &\leq 20 \\ -20 \leq X_2 &\leq 20, \end{aligned}$$

and graphical depictions of the feasible region and objective functions are shown in Figure 5.13. The Pareto optimal set and front, as published by Van Veldhuizen, are shown in Figures 5.14(a) and 5.14(b), respectively [148].

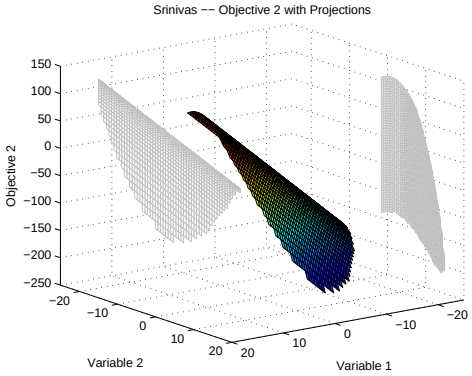
Implementations 6 and 7 were used to determine the experimental Pareto optimal set and front shown in Figures 5.14(c) and 5.14(d), respectively. As depicted in the graphs, the Pareto optimal set and front found via the SMOMADS algorithm match quite well to the published solution. In fact, the published solution, found via an evolutionary algorithm, clearly contains many dominated solutions, whereas the experimental solution contains very few.



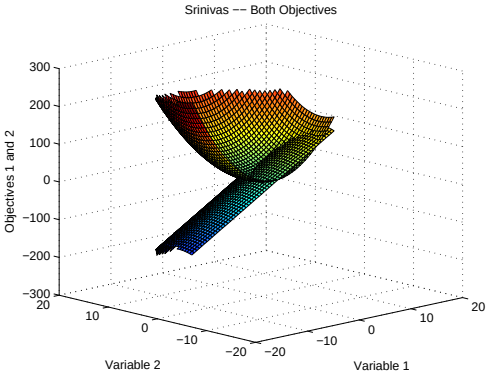
(a) Feasible Region



(b) Objective Function 1 with Projections onto the axes

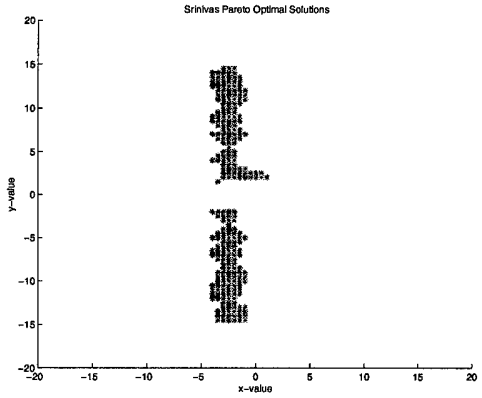


(c) Objective Function 2 with Projections onto the axes

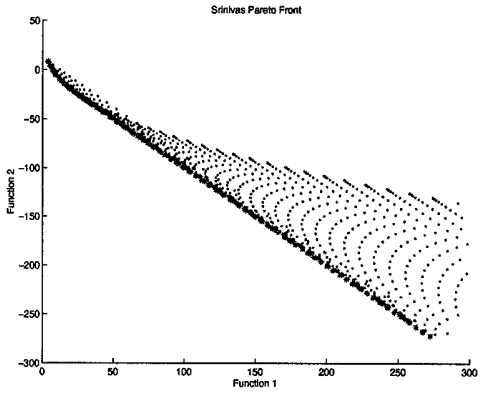


(d) Both Objective Functions

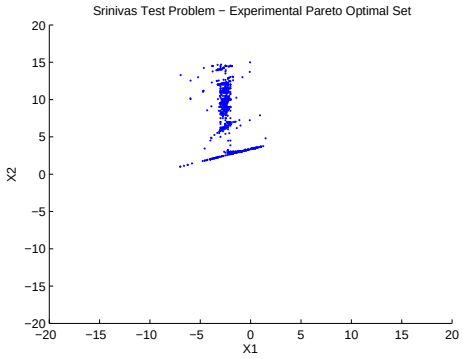
Figure 5.13: Graphical Depictions of the Srinivas Test Problem



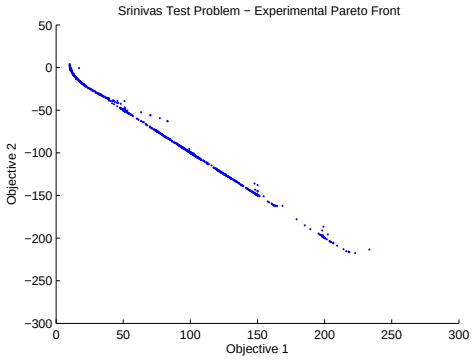
(a) Pareto Optimal Set for Deterministic Test Problem, Figure D.17 in [148]



(b) Pareto Front for Deterministic Test Problem, Figure D.18 in [148]



(c) Experimental Pareto Optimal Set



(d) Experimental Pareto Front

Figure 5.14: Comparison of Experimental Results to Published Solution for Srinivas Test Problem

5.1.2.6 DTLZ7 (2 Objective) Test Problem. The DTLZ7 test problem can be found in [119] and [41]. It is bi-objective with one linear and one nonlinear objective function and two decision variables with lower and upper bounds. The

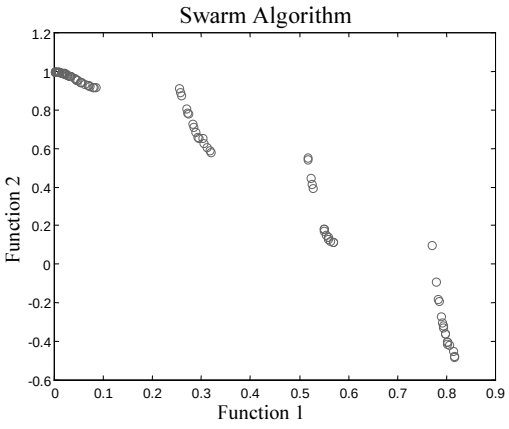
problem formulation is

$$\begin{aligned} \min \quad & f_1(X_1, X_2) = X_1 \\ & f_2(X_1, X_2) = (1 + 10X_2)\left(1 - \left(\frac{X_1}{1 + 10X_2}\right)^2 - \frac{X_1 \sin(8\pi X_1)}{1 + 10X_2}\right) \end{aligned}$$

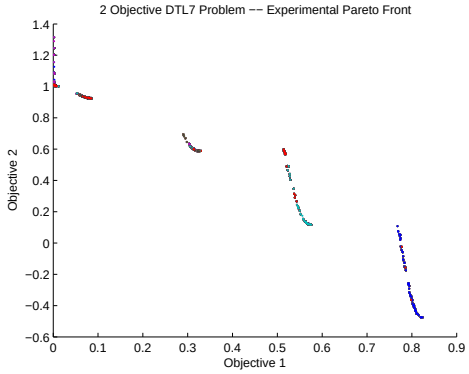
subject to

$$0 \leq X_1, X_2 \leq 1.$$

The pubished Pareto optimal front, as determined via a particle swarm algorithm [119], is shown in Figure 5.15(a). Implementations 2 and 4 were used to determine the experimental solution shown in Figure 5.15(b). When compared to the published solution, the experimental solution matches quite well.



(a) Pareto Front for Deterministic Test Problem, Figure 1 in [119]



(b) Experimental Approximation of the Pareto Front

Figure 5.15: Comparison of Experimental Results to Published Solution for 2 Objective DTLZ7 Test Problem

5.1.3 Mixed Variable Bi-Objective Test Problems.

5.1.3.1 Disk Brake Design Test Problem. This mixed variable test problem is found in [119] and is an engineering design problem for a disk brake. It contains two nonlinear objective functions, two linear constraints, three continuous

variables and one categorical variable, and is given by

$$\begin{aligned} \max \quad & f_1(X_1, X_2, X_3, X_4) = (4.9 \times 10^{-5})(X_2^2 - X_1^2)(X_4 - 1) \\ & f_2(X_1, X_2, X_3, X_4) = \frac{(9.82 \times 10^6)(X_2^2 - X_1^2)}{X_3 X_4 (X_2^3 - X_1^3)} \end{aligned}$$

subject to

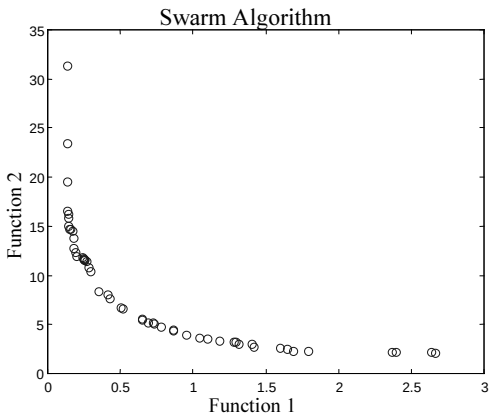
$$\begin{aligned} (X_2 - X_1) - 20 &\geq 0 \\ 30 - 2.5(X_4 + 1) &\geq 0 \\ 0.4 - \frac{X_3}{3.14(X_2^2 - X_1^2)} &\geq 0 \\ 1 - \frac{(2.22 \times 10^{-3}) X_3 (X_2^3 - X_1^3)}{(X_2^2 - X_1^2)} &\geq 0 \\ \frac{(2.66 \times 10^{-2}) X_3 X_4 (X_2^3 - X_1^3)}{(X_2^2 - X_1^2)} - 900 &\geq 0 \end{aligned}$$

$$\begin{aligned} 55 \leq X_1 \leq 80 \\ 75 \leq X_2 \leq 110 \\ 1000 \leq X_3 \leq 3000 \\ 2 \leq X_4 \leq 20 \\ X_4 \in \mathbb{N}. \end{aligned}$$

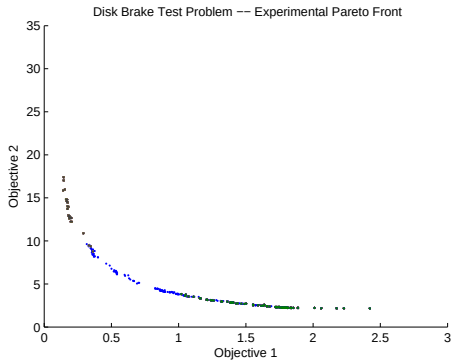
The lone categorical variable X_4 represents the number of disks in the brake. Though it can take on only 19 distinct values, meaning that exhaustive enumeration is not prohibitively expensive, the point of this test problem was to exercise the mixed variable logic in the solver. Implementation 7 of the SMOMADS algorithm was used to determine the Pareto front shown in Figure 5.16(b). When compared to the published solution (generated via a particle swarm algorithm) [119], the solution generated by SMOMADS is nearly identical.

5.2 Engineering Design Optimization Application

After the algorithm successfully solved the published test problems, it was applied to a small airfoil engineering design optimization problem. Mathematically, the



(a) Pareto Front for Deterministic Test Problem, Figure 6 in [119]



(b) Experimental Approximation of the Pareto Front

Figure 5.16: Comparison of Experimental Results to Published Solution for Disk Brake Test Problem

problem is:

$$\min_{\Delta M, \Delta\alpha} \left(E\left[c_d\right], \sigma^2\left(c_l\right), \sigma^2\left(c_d\right), -E\left[c_l\right] \right) \tag{5.12a}$$

where

$$\begin{aligned} c_l &= f_1(X_1, X_2, X_3) \\ c_d &= f_2(X_1, X_2, X_3) \end{aligned} \tag{5.12b}$$

subject to

$$\begin{aligned} E\left[c_l\right] &\geq 0.612 & \sigma^2\left(c_l\right) &\leq 0.0546 \\ E\left[c_d\right] &\leq 0.0172 & \sigma^2\left(c_d\right) &\leq 0.0061 \\ \left|E\left[c_m\right]\right| &\leq 0.0888 & \sigma^2\left(c_m\right) &\leq 0.00827 \\ 0 \leq X_1 < 10 \quad 0 \leq X_2 < 10 \quad 12 \leq X_3 < 100. & & & \end{aligned} \tag{5.12c}$$

This problem was adapted from one originally solved by Ciprian *et al.* [29] using multi-criteria decision making methods and game theory. The decision variables X_1 and X_2 represent the first and second digits in the NACA[3] airfoil number and X_3 represents

[3]NACA is the National Advisory Committee for Aeronautics, established by an act of the United States Congress on 3 March 1915 [7]. NACA ceased to exist on 1 October 1958 and was replaced by the National Aeronautics and Space Administration (NASA) [108]; however, the NACA numbering

the third and fourth digits. Following the standard NACA airfoil numbering system, this implies that X_1, X_2, and X_3 represent, respectively, the maximum airfoil camber (in percentage of its chord, the distance between the leading and trailing edges of an airfoil measured in the direction of normal airflow [156]), distance of maximum camber from the airfoil leading edge (in tens of percents of its chord), and the maximum thickness of the airfoil (in percent of its chord) [157]. As shown in Equation (5.12a), the objectives are to minimize the expected value and variance of the drag coefficient and the variance of the lift coefficient and to maximize[4] the expected value of the lift coefficient and the lift and drag coefficients are assumed to be black box functions of the decision variables. In the original problem, the coefficients of lift and drag are determined using the Navier-Stokes version of the flow solver MUFLO and AIRFOIL (using different shape coordinates as decision variables). Neither of these solvers were available open source, so the problem was modified slightly so that it could be used with the PABLO solver (which stands for Potential Flow around Airfoils with Boundary Layer Coupled One-way), a pedagogical low-speed airfoil analysis program written in MATLAB® [152]. The changes made to the design problem are as follows:

1. The original problem used two 10-degree Bèzier curves to represent the airfoil shape and used their coordinates as decision variables. The modified problem used the NACA airfoil numbering system to represent the airfoil shape and used the first, second, and third/fourth digits as decision variables.

2. The flow solver in the original problem used Mach number as an input. The solver in the modified version uses Reynolds number as an input. However, since the Reynolds number is directly proportional to velocity by

$$Re = \frac{\frac{\rho v_s^2}{L}}{\frac{\mu v_s}{L^2}} = \frac{\rho v_s L}{\mu} = \frac{v_s L}{v} \tag{5.13}$$

system continues to be used.

[4]The expected value of the lift coefficient is maximized by minimizing its negative.

where

υ_s - mean fluid velocity

L - characteristic length

μ - (absolute) dynamic fluid viscosity

ρ - fluid density

v - kinematic fluid viscosity: $v = \mu/\rho$

and

$$M = \frac{V}{a} = \frac{V}{\sqrt{\gamma RT}} \tag{5.14}$$

where

M - Mach number

V - velocity

γ - ratio of specific heats (for air $\gamma = 1.4$)

R - specific gas constant

T - temperature,

and for a fixed temperature and pressure, velocity can be determined from Mach number via Equation (5.14) [7, 158]. Unfortunately, the temperature, pressure, and characteristic length of the airfoil are not given in [29]; thus, standard temperature and pressure at sea level and the characteristic length of ten feet are assumed.

3. The original problem was near the transonic range, the range of speeds just below and above the speed of sound, approximately Mach 0.8 – 1.2. The flow solver in the modified problem is subsonic, and the speeds in original problem (mach 0.73 ± 0.05) approach the limits of a low-speed solver, so the speed range in the modified problem was reduced to mach 0.53 ± 0.05.

In both the original and modified problems, the objectives are to be minimized over a range of Mach numbers (Reynolds numbers) and angle of attack (2 ± 0.5) in order to find a robust design. (However, in the modified problem using the low-speed solver, the objectives do not vary across the range of Mach numbers, so the variation is due only to angle of attack.) The decision space is constrained such that the new airfoils can perform no worse than the RAE2822 airfoil[5] and the minimum airfoil thickness is 12% of the chord. (The performance values for the RAE2822 airfoil were taken from [29] and are given in Equation (5.12c)). This problem has been sufficiently modified such that direct comparison to the published solution is no longer germane and is not shown.

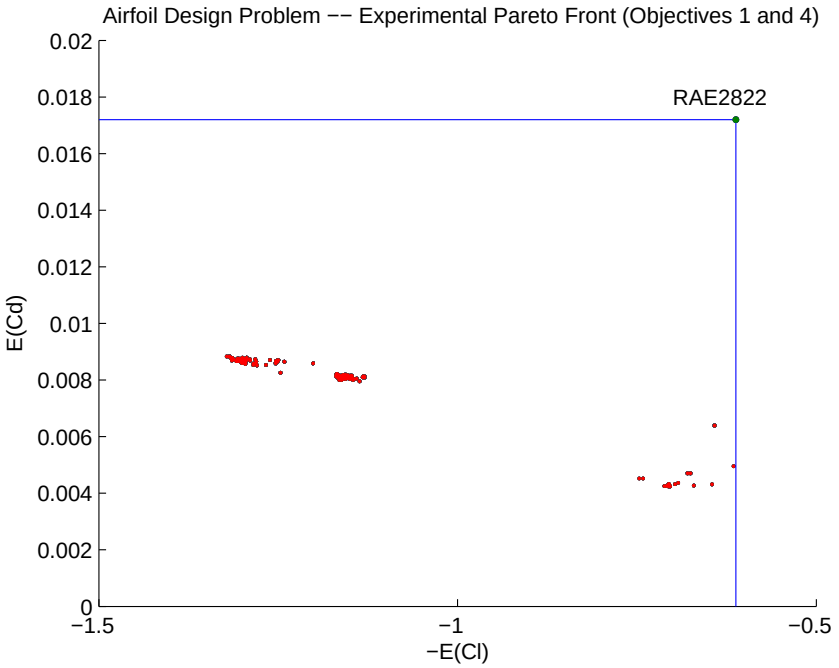
Implementation 7 of the SMOMADS algorithm was used to determine the experimental Pareto front, shown in Figure 5.17. In Figures 5.17(a) and 5.17(b), the feasible region is enclosed (below and to the left) by the solid lines.

Because the problem had to be modified to run inside the MATLAB® environment, comparisons to the published solution would be inappropriate. However, some observations regarding the experimental solution follow. The SMOMADS algorithm is able to find many non-dominated solutions, but each run took a significant amount of time (approximately 3 days for 100 design points). This underscores the necessity of a parsimonious experimental design in the master problem and highlights the need for follow-on research in this area.

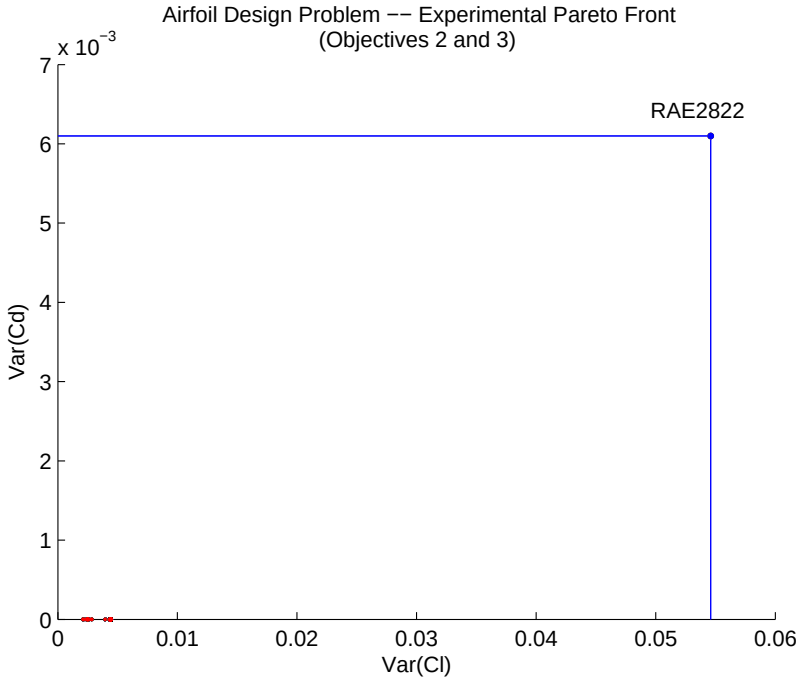
As with the test problems, the SMOMADS algorithm tended to find solutions in the "compromise" area of the Pareto front. This particular type of problem highlights the notion that such solutions are particular useful (see Section 4.4.2). For example, a Pareto optimal solution corresponding to high lift, but also high drag, may result in an aircraft that uses excessive amounts of fuel. Similarly, one with very low drag, but little lift, could result in an aircraft that cannot carry much weight. Thus, the solutions found by the algorithm would be those more likely to be favored by a

[5]RAE stands for the Royal Aircraft Establishment and is the British version of the United States' NACA airfoil system [7].

decision maker.



(a) Experimental Pareto Front for Objectives 1 and 4



(b) Experimental Pareto Front for Objectives 2 and 3

Figure 5.17: Experimental Results of the Airfoil Engineering Design Problem

5.3 Quality Metrics

The Pareto set quality metrics (Pareto spread, number of distinct choices, and cluster) discussed in Section 3.2.1.6 were calculated for all the test problems. The parameters of the test problems necessary for the calculations are given in Table 5.2 and the values of the test metrics are given in Table 5.3. Since, as previously mentioned, these metrics could be useful in determining when to terminate the master problem, the experimental Pareto set for each test problem was reduced by 90% by randomly selecting 10% of the Pareto optimal points and then reevaluating the metrics in order to demonstrate how they might be used with threshold values. The values of the metrics on the reduced sets are shown in Table 5.4, where it is evident that the test metrics do indeed improve as the number of points in the Pareto set increases.

Table 5.2: Parameters Required for the Quality Metrics

Problem	f_1^b	f_2^b	f_3^b	f_4^b	f_1^g	f_2^g	f_3^g	f_4^g	$\overline{np}$	$.1 \times \overline{np}$
Viennet 3	10	18	0.2		1	15	-0.2		3467	346
Viennet 4	7.5	-11	26		3.3	13	15		58990	5899
Poloni	30	50			0	0			35178	3417
Tamaki	0	0	0		-1	-1	-1		404	41
Dias Γ1	1	1			0	0			772	77
Dias Γ2	1.1	1.1			0	0			330	33
Fonseca F1	1.01	1.01			0	0			1066	107
Schaffer F3	1	16			-1	0			1958	196
Srinivas	250	10			0	-250			996	97
DTLZ7	0.85	1.4			0	-0.6			377	38
Disk Brake	2.75	33			0	0			289	29
Airfoil	0.0115	0.004	0.00025	-0.612	0	0	0	-2.5	231	24

Because the sets were reduced in a random fashion, the quality of the sets of the different test problems were not reduced equally, meaning that improvement is more significant in some cases than in others. This also provides some insight into the utility of the metrics. If the quality metrics are tracked after each design point, they could, in fact, be used as stopping criteria, because relatively few (quality) points can produce good values in the test metrics. For example, in the Viennet 3 and Viennet 4 test problems, values for Pareto spread corresponding to the the reduced Pareto

optimal set are nearly as high as those for the full set. If this had been the first points generated, the algorithm could have been terminated after only 10% of the experimental design had been evaluated. Further research in this area is suggested in Section 6.2.3.2.

Table 5.3: Quality Metrics Applied to Test Problems

Problem	$OS(P)$	$OS_1(P)$	$OS_2(P)$	$OS_3(P)$	$OS_4(P)$	$NDC_\mu(P)$	$CL_\mu(P)$
Viennet 3	0.4496	0.9072	0.7010	0.7070		96	36.1
Viennet 4	0.5524	0.9797	0.5750	0.9806		173	344.8
Poloni	0.2054	0.3660	0.5613			53	666.7
Tamaki	0.5304	0.8287	0.8535	0.7499		344	1.2
Dias Γ1	0.6987	0.8013	0.8719			125	6.2
Dias Γ2	0.8619	0.9182	0.9388			61	5.4
Fonseca F1	0.9942	0.9975	0.9968			150	7.1
Schaffer F3	0.9933	0.9936	0.9997			98	20.0
Srinivas	0.7614	0.8929	0.8528			155	6.4
DTLZ7	0.8668	0.9675	0.8959			76	5.0
Disk Brake	0.3830	0.8294	0.4618			87	3.3
Airfoil	0.1528	0.4415	0.4462	0.8911	0.8703	43	5.4

Table 5.4: Quality Metrics Applied to Reduced Solution Sets of the Test Problems

Problem	$OS(P)$	$OS_1(P)$	$OS_2(P)$	$OS_3(P)$	$OS_4(P)$	$NDC_\mu(P)$	$CL_\mu(P)$
Viennet 3	0.4296	0.8977	0.6790	0.7048		42	8.7
Viennet 4	0.4245	0.9303	0.5496	0.8302		136	43.3
Poloni	0.0164	0.2925	0.0559			36	98.0
Tamaki	0.0211	0.3105	0.3025	0.2249		40	1.0
Dias Γ1	0.6908	0.7951	0.8689			54	1.4
Dias Γ2	0.8132	0.8831	0.9209			19	1.7
Fonseca F1	0.9927	0.9975	0.9952			57	1.9
Schaffer F3	0.9796	0.9803	0.9992			69	2.8
Srinivas	0.5817	0.7668	0.7586			46	2.1
DTLZ7	0.6557	0.9396	0.6978			20	1.9
Disk Brake	0.3830	0.8294	0.4618			26	1.1
Airfoil	0.0175	0.3671	0.4462	0.8897	0.1203	14	1.7

5.4 Efficiency of the Algorithm

The computational efficiency of SMOMADS is dependent on several factors, which are determined by choices made during implementation of the algorithm. Dis-

cussion of these factors and observations from the test problems follow.

5.4.1 Experimental Design. In the master problem, the choice of experimental design directly affects the computation time required by algorithm. For example, in the first test problem (Viennet 4), the FFD was chosen, resulting in 4,096 individual subproblems, with 5 replications. Computational time for each replication took considerably less than a minute, but total computational time was several hours. For problems with expensive or lengthy function evaluations, such a design would be prohibitively expensive. This was highlighted by subsequent test problems. For example, the FFD chosen for the Poloni test problem resulted in 10,000 individual subproblems with 5 replications. The computational time required was over 72 hours. If a CCD had been chosen instead, the number of subproblems required would have been reduced by 3 orders of magnitude; therefore, this design was used in later test problems in a sequential experimentation scheme (see Section 4.1). The number of test points for each problem and whether each problem was solved using sequential experimentation are shown in columns 6 and 5 of Table 5.1, respectively. Every test point was replicated 5 times, except for the first CCD (36 test points) of the disk brake problem that was replicated 15 times. Further research regarding experimental designs is suggested in Section 6.2.3.

5.4.2 Termination Criteria. The overall computational time required by SMOMADS is heavily dependent on the computational efficiency of the subproblems[6]. The computation time required by the subproblems depends on algorithmic parameters of the two individual solvers. These parameters were set to suggested default values [3], with the exception of the maximum number of iterations allowed in the subproblem, which was set to 500. Limiting the number of iterations increased the computational efficiency by stopping subproblems that might be converging at a slower rate and thereby substantially increasing the overall computation time. In

[6]The number of objective function evaluations required by the solvers is not captured with current code and could not be reported. Modifications to the code are suggested in Section 6.2.4.1

addition to directly impacting computation time, limiting iterations also indirectly impacts computational efficiency as described in the next section.

5.4.3 Solution Quality and Filtered Points. In the limit, the subproblems always converge to Pareto stationary points; however, as discussed in Section 3.2.1.5, with a finite number of iterations, the solver may terminate at a dominated point. Thus, if a maximum number of iterations is stipulated, it increases the likelihood of premature termination and directly affects solution quality in the subproblems. Dominated points are filtered and not allowed to enter the efficient set, so the final solution quality is not affected; however, computational effort is wasted for each point that must be discarded. Therefore, the maximum number of iterations must be set low enough that dubious subproblems are abandoned, yet not so low that subproblems are squandered. Unfortunately, this can be problem-specific. For example, in the Viennet 3 test problem, 83% of points were filtered, but in the Poloni test problem, only 30% of solutions did not enter the efficient set.

5.5 Summary

In this chapter, the accuracy of the SMOMADS algorithm was verified by its use on a set of test problems with known solutions and an airfoil design problem. The algorithm generally produces numerical results quite similar to those in the literature. In the design problem, it was able to generate many Pareto optimal points. Additionally, factors affecting computational efficiency were discussed and applicable examples from the test problems were illustrated.

VI. Conclusions and Recommendations

A research approach extending stochastic and multi-objective solution methods to one applicable to optimization problems having both characteristics has been presented. The new method further extends the generalized pattern search methods that previously existed for single-objective optimization of mixed-variable systems having stochastic responses to the multi-objective case. Such problems are typically encountered when one desires to optimize systems with multiple, often competing, objectives that do not have a closed form representation and must be estimated via simulation. A two-stage method was presented that combines generalized pattern search/ranking and selection (GPS-R&S) developed for single-objective stochastic problems and Mixed Variable Mesh Adaptive Direct Search (MVMADS) with three multi-objective methods: interactive techniques for the specification of aspiration/reservation levels, scalarization functions, and multi-objective ranking and selection.

Convergence analysis was conducted showing that in each subproblem, a subsequence of iterates converges with probability one to a stationary point appropriately defined in the mixed-variable domain, and that if that solution is also optimal, then it is Pareto optimal. Algorithmic implementations of the first stage have been developed and computational experiments conducted on standard multi-objective test problems with random noise added to the objective function evaluations to simulate measurement error or the use of a simulation. Finally, the algorithm was applied to an aeronautical engineering design optimization problem. The original contributions of this dissertation research are summarized in Section 6.1 and future research directions are proposed in Section 6.2.

6.1 Contributions

This research provided the following original contributions to the field of Operations Research.

6.1.1 General Solution Method for Multi-Objective Optimization of Mixed Variable Systems having Stochastic Responses. The primary contribution of this research is the development of the first provably convergent algorithm for solving multi-objective, stochastic optimization problems over a mixed-variable domain. Though other methods exist for problems having one or two of these characteristics, no existing method was applicable to those problems exhibiting all three.

6.1.2 Convergence Analysis. The second contribution of this research is a rigorous convergence analysis of the algorithm. The new algorithm was shown to converge almost surely to stationary points appropriately defined in the mixed-variable domain. It was further shown that if such points are optimal, they are also Pareto optimal. In proving convergence, it was necessary to fill a void in the existing convergence analysis for MADS, *i.e.*, to show that a mixed-variable version of MADS converges almost surely to stationary points appropriately defined in the mixed variable case.

6.1.3 Algorithmic Implementation Developed and Testing. The third contribution of this research was the development of seven implementations of the algorithm that were tested on multi-objective optimization problems from the literature. These problems had random noise introduced into their objective function evaluations to simulate random responses. Even in the presence of noise, the algorithm was shown to produce results comparable to the published solutions that were generated with deterministic objective function evaluations.

6.1.4 Application Heuristic Stopping Criteria. Heuristic stopping criteria were developed for the portion of the algorithm referred to as the master problem. An iterative process was presented (see Section 4.4.2) for determining when an adequate approximation to the Pareto set had been found. Metrics were suggested to substantiate the quality of the set and for termination.

6.2 Future Research

6.2.1 Convergence Results for MVMADS-RS. Convergence results for the stochastic, multi-objective, nonlinearly constrained case have not been rigorously proven and depend on Conjecture 3.3.10. A formal proof of this conjecture is required for completeness.

6.2.2 Stage 2. As discussed in Section 3.2.2, some efficient points may not be found via interactive specification of aspiration/reservation levels and scalarization functions of the type discussed in Section 3.1.4. Thus, a second (optional) stage was suggested for those cases in which missing points may pose a particular problem. In this stage, a discrete mesh (similar to that used for MGPS-RS and MADS) around the current efficient points would be determined and then a multi-objective ranking and selection algorithm would be used to check for new efficient points on the mesh. In this research, the algorithm performed so well on the test problems without the second stage, that adding it was not necessary; however, future research is suggested for those cases in which it may be required.

6.2.3 Search Methods in the Objective Space. Though the proposed method is applicable to problems with any number of objectives, in practice, the dimensionality in the experimental design limits searching in the objective space. Thus, an efficient manner of searching the objective space is needed, and the following possible research areas are suggested.

6.2.3.1 Response Surface Methodology. As suggested in Section 4.1, the problems studied here can be thought of as approximating a response surface (the Pareto front) with aspiration and reservation levels as the decision variables using experimental design methods. In this research, only three experimental designs were studied: full factorial design, central composite design, and Box-Behnken design. Other methods and principles from experimental design may be as useful in efficiently approximating the Pareto front. Thus a more in depth study of these methods is

suggested.

6.2.3.2 Quality Metrics for the Pareto Set. Currently, the experimental design in the master problem of the solution algorithm is run exhaustively. In some cases though, an adequate Pareto front approximation may be obtained before the entire design is completed. Thus, if the quality of the experimental Pareto set could be evaluated at various points and the search terminated at a predetermined threshold, the efficiency of the search method could be improved.

6.2.3.3 Improved Filter. The filter applied in this research used a simple comparison of objective function values to determine if a point was dominated. Because of the stochastic nature of the problem, this could potentially lead to some dominated points entering the set or non-dominated points being unduly excluded. The experimental results in Section 5.1 indicated that error in the efficient set when compared to known results was insignificant; however, it is possible that in other optimization problems, such a filter would not be adequate. One suggestion is to add a multi-objective ranking and selection [85] algorithm, in place of the simple comparison currently in the filter to determine if a point is dominated (see Sections 3.2.2.1 and 3.1.5).

Additionally, the filter only determines if a point is dominated before it enters the efficient set. Once in the set, it is never again considered. It is possible that an erroneous point could enter the efficient set early in the algorithm's run. Although the experimental results in this study did not indicate significant problems with solution accuracy, it is recommended that the filter also determine if an entering efficient point dominates any existing efficient solutions.

6.2.3.4 Master Problem Termination Criteria. Though heuristic stopping criteria have been developed here (see Section 4.4.2), they could be refined based on response surface methodology (Section 6.2.3) and quality metrics (Section 6.2.3.2). With an astute experimental design, a more thrifty search of the objective space could

be conducted. Furthermore, with insightful quality metrics applied after each design point is evaluated, the search could be stopped when a desired quality level is reached, rather than exhaustively evaluating the experimental design points. With both elements, the search could be conducted as parsimoniously as possible.

6.2.3.5 Automated Decision Agent. Section 3.2.1.1 discusses the experimental design built to investigate a range of values of aspiration and reservation levels. If, instead, an automated decision agent (see Sections 2.1.1.2 and 2.3.1) could be developed, it may provide better insight to the decision maker. In fact, even the decision strategies of a decision maker could be investigated, *e.g.*, conservative versus daring decision strategies.

6.2.4 Software Changes.

6.2.4.1 Code to Capture Metrics. Though observations have been made about solution quality and computational efficiency, because existing solvers were used, some metrics were not captured and could not be reported. The following changes to the software are suggested.

1. As discussed in Section 4.4.1, in MGPS-RS, the difference between the response standard deviation S and the indifference zone parameter is used in the termination criteria for the algorithm; however, this data is not reported at the end of the subproblem. (Response standard deviation in this case is that of the scalarized objective function, not the individual objective functions.) It is suggested that the response standard deviation at the end of each subproblem be captured. Though not a direct measure of the variance in each objective function, it does provide insight into the variability of the approximate Pareto set.
2. Though computational efficiency of the algorithm was discussed in Section 5.4, some metrics typically associated with computational efficiency were not cap-

tured by the existing software and could not be reported. Such measures would be helpful in further assessing the algorithm's potential use on computationally expensive problems. Specifically, it is suggested that the software be modified to capture CPU (central processing unit) time used by each subproblem, overall CPU time, number of iterations in each subproblem, and number of objective function evaluations.

6.2.4.2 User-Friendly Software. This research provided the algorithmic methodology for a two-stage solution process and prototype code for specific implementations of the algorithm. This code was integrated with the batch mode of NOMADm, but was not integrated with the graphical user interface (GUI). In future research, it is suggested that the code controlling the aspiration/reservation level analysis be integrated with NOMADm into a single graphical user interface. Additionally, this integrated code should then be compiled using MATLAB®'s VB.net®s builder toolbox so that it can be executed by users without having to have a MATLAB® license.

1.1 File name: DiasGamma2RuntheBatchFile.m

```
 SetUpProblem
 global numobjectives;
 global aspire;
 global reservation;
 global utopia;
 global nadir;
 global objfunctionstruc;
 global xvals;


%Define problem files
   MyProb.problemPath = fullfile(matlabroot,...
           'work','examples','theprob','ContBOIcky','DiasGamma2');
   MyProb.F = 'DiasGamma2';             % functions file
   MyProb.O = 'DiasGamma2_Omega';       % linear constraints file
   MyProb.X = 'DiasGamma2_X';           % closed constraints file
   MyProb.I = 'DiasGamma2_x0';          % initial points file
   MyProb.N = 'DiasGamma2_N';           % discrete neighbor file
   MyProb.P = 'DiasGamma2_Param';       % parameter file
   MyProb.C = 'DiasGamma2_Cache.mat';   % previously created Cache file
   MyProb.S = 'DiasGamma2_Session.mat'; % previously created Session file
   MyProb.H = 'DiasGamma2_History.txt'; % iteration history text file
   MyProb.D = 'DiasGamma2_Debug.txt';   % debug log file
   MyProb.fType  = 'M';                 % type of functions file {M,F,C}
   MyProb.nc     = 0;                   % number of nonlinear constraints
%other variables
numobjectives = 2; utopia(1) = 0; utopia(2) = 0; nadir(1) = 1;
```

```
nadir(2) = 10;
%range checked
 arange=[.1 .8; .05 1.];
 rrange=[.81 1.2; 1.1 8];

numaspire=5; numreservation=5; numreplications=5;

%Types of designs 'fullfactorial', 'centralcomposite', 'boxbehnken'
%You need the statistics toolbox for this to work
%Need file generatepoints.m in your workpath
TestPoints = generatepoints('fullfactorial',numaspire,
numreservation, arange, rrange, numobjectives); MyResult =[];
objfunctionstruc = []; bogus=0; xvals = []; for
Mym=1:size(TestPoints,1)
    for Myn=1:numobjectives
        aspire(Myn) = TestPoints(Mym,Myn);
        reservation(Myn) = TestPoints(Mym,numobjectives+Myn);
    end
    if (mod(Mym,50)==0)
        Mym
    end
    for l = 1:numreplications
        temp = size(MyResult,1)+1;
        %This is where the MADS or GPS solver is called.
        %Need the MOmads.m file installed in the workpath.
        %For MOmads, you need the
        %SetUpProblem.m and nomad.m files also.

        temp1 = MOmads(MyProb,Options);
```

```
        temp2=temp1.param;
        if temp == 1
            AddYes = 1;
        else
            AddYes = CheckPareto(temp2, MyResult, numobjectives);
        end
        if AddYes == 1
            MyResult(temp,1).x = temp1.x;
            MyResult(temp,1).f = temp1.f;
            MyResult(temp,1).c = temp1.c;
            MyResult(temp,1).aspire=aspire;
            MyResult(temp,1).reservation=reservation;
            MyResult(temp,1).Objective=temp2;
        else
            bogus = bogus+1;
        end
    end
end

%This section was for graphing the output.  Uncomment as needed.

% for m=1:size(MyResult,1)
%         MyX(m)=MyResult(m,1).x(1);
%         MyY(m)=MyResult(m,1).x(2);
% end
%         scatter(MyX,MyY,3,'filled')
% clear MyX MyY;
% figure
```

```
% hold
% for m=1:size(MyResult,1)
%         MyX(m)=MyResult(m,1).Objective(1);
%         MyY(m)=MyResult(m,1).Objective(2);
% end
% scatter(MyX,MyY,3,'filled')
% clear MyX MyY;
% figure
% hold
% clear MyX MyY;
% figure
% hold
% for m=1:size(MyResult,1)
%         MyX(m)=MyResult(m,1).Objective(2);
%         MyY(m)=MyResult(m,1).Objective(3);
%         scatter(MyX,MyY,3,'filled')
% end
% clear MyX MyY;
```

1.2 File name: DiasGamma2.m

```
function [fx,cx] = DiasGamma2(x);
 global numobjectives;
 global aspire;
 global reservation;
 global utopia;
 global nadir;
 global objfunctionstruc;
 global xvals;

%f(1) and f(2) are the objective functions.
f(1)=x(1)+.01*rand; g1=sum(x)-x(1); g2=1+(9/29)*g1;
f(2)=g2*(1-(f(1)/g2)^2)+0.01*rand;

%This saves the original function values to the variable param.
%Used in the batch file for the filter.
Param.param=f;

%set epsilon and alpha and beta and w
epsilon = 5;

%This code puts the objectives into the achievement
% scalarization function.
for n1 = 1:numobjectives
  w(n1)= 1/(reservation(n1)-aspire(n1));
if (aspire(n1) ~= utopia(n1))
  alpha(n1)=0.1*(reservation(n1)-aspire(n1))/(aspire(n1)-utopia(n1));
else
  alpha(n1)=0.1*(reservation(n1)-aspire(n1))/(0.0000001);
```

```
end if (aspire(n1) ~= nadir(n1))
  beta(n1)=-10*(reservation(n1)-aspire(n1))/(aspire(n1)-nadir(n1));
else
  beta(n1)=-10*(reservation(n1)-aspire(n1))/(0.0000001);
end end for n2 = 1:numobjectives
  if f(n2) < aspire(n2)
      u(n2)=alpha(n2)*w(n2)*(aspire(n2)-f(n2))+1;
  elseif f(n2)<=reservation(n2)
      u(n2)=w(n2)*(aspire(n2)-f(n2))+1;
  else
      u(n2)=beta(n2)*w(n2)*(reservation(n2)-f(n2));
  end
end temp = [u(1) u(2)];

%This is the combined objective function.
fx = -(min(temp)+epsilon*sum(temp));

%This would be used if you had non-linear constraints.
cx=[];

%Don't delete this line or the param variable above won't save.
setappdata(0,'PARAM',Param);

return
```

1.3 File name: DiasGamma2_Omega.m

```
%*********************************************************************
%canoeDW_Omega: User-supplied funct. for defining Omega, based on p.
%*********************************************************************
function [A,l,u,plist] = DiasGamma2_Omega(n); A = eye(n); l = [0; 0;
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; ...
    0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0];
u = [1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1;...
    1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
return
```

1.4 File name: DiasGamma2_x0

```
function iterate = DiasGamma2_x0;
%This sets the initial points.
iterate(1).x = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; ...
    0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0];
iterate(1).p = {}; return;
```

1.5 File name: SetupProblem.m

```
%*********************************************
% DO NOT MODIFY THIS SECTION
%*********************************************

% Set Options to their default values
clc clear variables Defaults = mads_defaults('Truth'); Options  =
Defaults.Options;

%************************************************
%Options  MAKE CHANGES HERE FOR SEARCH TYPE, ETC.
%************************************************
% Specify Choices for SEARCH
Options.nSearches         = 2;
Options.Search(1).type    = 'LHS';% For choices, see mads_defaults
Options.Search(1).nIter   = 1;    % Number of iter for Search #1
Options.Search(1).nPoints = 8;    % Number of poll or sample points
Options.Search(1).sfile   = '';   % filename must include full path
Options.Search(1).file    = '';   % filename must include full path
Options.Search(1).local   = 0;    % flag to turn on trust region
Options.Search(1).merit   = 0;    % flag to penalize clustered data
Options.Search(1).param   = 0;    % flag to constr. surr. on param
Options.Search(1).recal = ...
strncmp(Options.Search(1).type(max(1,end-1):end), ...
  'NW',2) || strncmp(Options.Search(1).type(max(1,end-3):end),'DACE',4);
Options.Search(2).type    = 'None';    % For choices, see mads_defaults
Options.Search(2).nIter   = 10;        % Number of iter for Search #2
Options.Search(2).nPoints = 1;         % Number of poll or sample points
Options.Search(2).sfile   = 'regpoly0';% filename must include full path
```

```
Options.Search(2).file    = '';        % filename must include full path
Options.Search(2).local   = 0;         % flag to turn on trust region
Options.Search(2).merit   = 0;         % flag to penalize clustered data
Options.Search(2).recal = ...
strncmp(Options.Search(1).type(max(1,end-1):end), 'NW',2) || ...
   strncmp(Options.Search(1).type(max(1,end-3):end), 'DACE',4);
Options.SurOptimizer      = 'mads'; Options.mvp1Surrogate     = 1;

Options.dace(2).reg       = 'regpoly0'; Options.dace(2).corr      =
'correxp'; Options.dace(2).theta     = 1; Options.dace(2).lower
= 0.01; Options.dace(2).upper     = 1000; Options.dace(2).isotropic
= 0;

% Specify Choices for POLL
Options.pollStrategy    = 'MADS_2n'; % For choices, see mads_defaults
Options.pollOrder       = 'Consecutive'; % For choices, see defaults
Options.pollCenter      = 0;         % Poll around n-th filter point
Options.pollComplete    = 0;         % Flag for complete polling
Options.NPollComplete   = 0;         % Flag for complete neigh. polling
Options.EPollComplete   = 0;         % Flag for complete extend polling

% Specify Termination Criteria
Options.Term.delta      = 1e-4;      % minimum mesh size
Options.Term.nIter      = Inf;       % maximum number of iterations
Options.Term.nFunc      = 50000;     % maximum number of function evals
Options.Term.time       = Inf;       % maximum CPU time
Options.Term.nFails     = Inf;       % max number of consec. Poll fails

% Choices for Mesh Control
```

```
Options.delta0          = 1;         % initial mesh size
Options.deltaMax        = 1;         % bound on how coarse mesh can get
Options.meshRefine      = 0.5;       % mesh refinement factor
Options.meshCoarsen     = 2.0;       % mesh coarsening factor

% Choices for Filter management (for problems with nl constraints)
Options.hmin            = 1e-8;      % minimum infeasible point h-value
Options.hmax            = 1.0;       % maximum h-value of a filter point

% Choices for EXTENDED POLL (for MVP problems)
Options.ePollTriggerF   = 0.01;      % f-value Extended Poll trigger
Options.ePollTriggerH   = 0.01;      % h-value Extended Poll trigger

% MADS flag parameter values
Options.loadCache        = 0;        % load pre-existing Cache file
Options.countCache       = 0;        % count Cache points as func. calls
Options.runStochastic    = 1;        % runs problem as a stoch. problem
Options.scale            = 2;        % scale directions using log base
Options.useFilter        = 1;        % filter(0=none,1=multi-pt,2=2-pt)
Options.degeneracyScheme = 'random'; % scheme for degenerate constraints
Options.removeRedundancy = 1;        % discard redundant lin. constr.
Options.computeGrad      = 0;        % compute gradient, if available
Options.saveHistory      = 0;        % saves MADS perform. to text file
Options.plotHistory      = 0;        % plot MADS performance
Options.plotFilter       = 0;        % plot the filter real-time
Options.plotColor        = 'k';      % color of history plot
Options.debug            = 0;        % turn on status mess for debugging
Options.showFilterPlot   = 0;        % turn off the filter plot
```

```
Options.Sur.Term.delta   = 0.01;     % surrogate minimum mesh size

% Set up figure handles for real-time plots

if (Options.plotHistory == 2)
   figure;
   Options.hplothandle = gca;
end
```

1.6 File name: generatepoints.m

```
function [TestPoints] = generatepoints(designofexperiments,
numasp,...
    numres, arange, rrange, numobj)
switch lower(designofexperiments)
    case 'fullfactorial'
        gen = [];
        for ar=1:2
            for n=1:numobj
                if ar==1
                    gen = [gen numasp];
                else
                    gen = [gen numres];
                end
            end
        end
        myfact=fullfact(gen);
        testpt=myfact;
        for m=1:size(myfact,1)
            for l=1:size(myfact,2)
                if l<=numobj
                    testpt(m,l)=arange(l,1)+(myfact(m,l)-1)*...
                        ((arange(l,2)-arange(l,1))/(numasp-1));
                else
                    testpt(m,l)=rrange(l-numobj,1)+(myfact(m,l)-1)...
                        *((rrange(l-numobj,2)-...
                        rrange(l-numobj,1))/(numres-1));
                end
            end
```

```
    end
    TestPoints=testpt;
case 'centralcomposite'
    myfact=ccdesign(2*numobj);
    testpt=myfact;
    for m=1:size(myfact,1)
        for l=1:size(myfact,2)
            if l<=numobj
                testpt(m,l)=0.5*((arange(l,2)+arange(l,1))+...
                    (myfact(m,l)*(arange(l,2)-arange(l,1))));
            else
                testpt(m,l)=0.5*((rrange(l-numobj,2)+...
                    rrange(l-numobj,1))+(myfact(m,l)*...
                    (rrange(l-numobj,2)-rrange(l-numobj,1))));
            end
        end
    end
    TestPoints=testpt;

case 'boxbehnken'
    myfact=bbdesign(2*numobj);
    testpt=myfact;
    for m=1:size(myfact,1)
        for l=1:size(myfact,2)
            if l<=numobj
                testpt(m,l)=0.5*((arange(l,2)+arange(l,1))+...
                    (myfact(m,l)*(arange(l,2)-arange(l,1))));
            else
                testpt(m,l)=0.5*((rrange(l-numobj,2)+...
```

```
                        rrange(l-numobj,1))+(myfact(m,l)*...
                        (rrange(l-numobj,2)-rrange(l-numobj,1))));
                end
            end
        end
        TestPoints=testpt;
    case 'singlepoint'
        testpt=[];
        for n=1:2*numobj
            if n<=numobj
                testpt=[testpt arange(n,1)];
            else
                testpt=[testpt rrange(n,1)];
            end
        end
        TestPoints=testpt;
    otherwise
        disp('I need a valid type of design of experiments.')
end
```

1.7 File name: CheckPareto.m

```
function YesNo = CheckPareto(functionvalues, MyResult,
numobjectives) thisone=[];

for m=1:numobjectives
        thisone = [thisone functionvalues(1,m)];
end for n=1:size(MyResult,1)
    thatone = [];
    for m=1:numobjectives
         thatone = [thatone MyResult(n,1).Objective(m)];

    end
    MyCheck=sum(sum(thatone < thisone));
    if MyCheck == numobjectives
        YesNo=0;
        break
    end
    YesNo=1;
end
```

1.8 File name: MOmads.m

```
function [BestF,BestI,RunStats,RunSet] = MOmads(MyProb,Options)
%MADS_BATCH  Sets up and runs the MADS algorithm without a GUI.
%
problemPath = MyProb.problemPath;
Problem.File.F = MyProb.F;        % functions file
Problem.File.O = MyProb.O;        % linear constraints file
Problem.File.X = MyProb.X;        % closed constraints file
Problem.File.I = MyProb.I;        % initial points file
```

```
Problem.File.N = MyProb.N;        % discrete neighbor file (MVP only)
Problem.File.P = MyProb.P;        % parameter file
Problem.File.C = MyProb.C;        % previously created Cache file
Problem.File.S = MyProb.S;        % previously created Session file
Problem.File.H = MyProb.H;        % iteration history text file
Problem.File.D = MyProb.D;        % debug log file
Problem.fType  = MyProb.fType;    % type of functions file {M,F,C}
Problem.nc     = MyProb.nc;       % number of nonlinear constraints
Problem.nameCache   = 'CACHE'; Problem.typeProblem = 'TRUTH';

%
%*********************************************************************
% DO NOT MODIFY AFTER THIS POINT
%*********************************************************************
if (Problem.nc == 0)
   Options.plotFilter = 0;
end if (Options.plotFilter) && (Options.showFilterPlot == 1)
   figure;
   Options.fplothandle = gca;
end

% Set the path, and load any user-provided problem parameters
cwd = pwd;           %create variable to remember current directory
cd(problemPath);     %change focus to problem directory
if (exist(Problem.File.P,'file') == 2)
   Problem.Param = feval(Problem.File.P);
   setappdata(0,'PARAM',Problem.Param);
end
```

```
% Get the initial iterates and call the optimizer
if isfield(Problem,'Param') && isfield(Problem.Param,'iterate0')
   iterate0 = Problem.Param.iterate0;
else
   iterate0 = feval(Problem.File.I);
end

[BestF,BestI,RunStats,RunSet] = mads(Problem,iterate0,Options);


cd(cwd);                %change focus back to original directory
return
```

Bibliography

1. Abraham, A. and Lakhmi, J. (2005). *Evolutionary Multiobjective Optimization*. Springer-Verlag, London.

2. Abramson, M. A. (2002). *Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems*. Ph.D. thesis, Rice University, Department of Computational and Applied Mathematics (CAAM). Also available as Rice CAAM Technical Report TR02-11.

3. Abramson, M. A. (2006). NOMADm version 4.02. Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. Available for download at http://www.afit.edu/en/ENC/Faculty/MAbramson/NOMADm.html.

4. Abramson, M. A., Asaki, T. J., Dennis, Jr., J. E., O'Reilly, K. R., and Pingel, R. L. (2007). Quantitative object reconstruction using Abel transform x-ray tomography and mixed variable optimization. Technical Report #TR07-03, Rice University, Department of Computational and Applied Mathematics (CAAM).

5. Abramson, M. A., Audet, C., Chrissis, J. W., and Walston, J. G. (2007). Mesh adaptive direct search algorithms for mixed variable optimization. Technical Paper GERAD G-2007-47, GERAD and Department of Mathematics and Industrial Engineering, École Polytechnique, Montréal, Canada.

6. Abramson, M. A., Audet, C., and Dennis, Jr., J. E. (to appear). Filter pattern search algorithms for mixed variable constrained optimization problems. *Pacific Journal of Optimization*. Also appears as Technical Report TR04-09, Department of Computational and Applied Mathematics, Rice University, Houston, Texas, 2004.

7. Anderson, Jr., J. D. (1989). *Introduction to Flight*. McGraw-Hill, Inc., New York, NY, third edition.

8. Anderson, Jr., J. D. (2000). *Introduction to Flight*. McGraw-Hill, Inc., New York, NY, fourth edition.

9. Audet, C. and Dennis, Jr., J. E. (2000). Pattern search algorithms for mixed variable programming. *SIAM Journal on Optimization*, **11**(3), 573–594.

10. Audet, C. and Dennis, Jr., J. E. (2004). A pattern search filter method for nonlinear programming without derivatives. *SIAM Journal on Optimization*, **14**(4), 980–1010.

11. Audet, C. and Dennis, Jr., J. E. (2006). Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, **17**(1), 188–217.

12. Audet, C., Savard, G., and Zghal, W. (2006). Multiobjective optimzation through a series of single objective formulations. Technical Paper GERAD G-2007-05, GERAD and Department of Mathematics and Industrial Engineering, École Polytechnique, Montréal, Canada.

13. Baba, N. and Morimoto, A. (1990). Three approaches for solving the stochastic multi-objective programming problem. In *Proceedings of GAMM/IFIP-Workshop on Stochastic Optimization*. Springer-Verlag, New York, NY, USA.

14. Baba, N. and Morimoto, A. (1993). Stochastic approximation method for solving the stochastic multi-objective programming problem. *International Journal of Systems Science*, **24**(4), 789–796.

15. Barto, A. G. (1992). *Reinforcement Learning and Adaptive Critic Methods*. Van Nostrand Reinhold, New York, NY, 469–491.

16. Barton, R. and Ivey Jr., J. (1996). Nelder-Mead simplex modifications for simulation optimization. *Management Science*, **42**(7), 954–973.

17. Beal, J. M., Shukla, A., Brezhneva, O. A., and Abramson, M. A. (to appear). Optimal sensor placement for enhancing sensitivity to change in stiffness for structural health monitoring. *Optimization and Engineering*.

18. Beausoleil, R. (2001). Multiple criteria scatter search. In *Proceedings of 4th Metaheuristics International Conference*. 539–543.

19. Beausoleil, R. (2006). MOSS multiobjective scatter search applied to non-linear multiple criteria optimization. *European Journal of Operational Research*, **169**, 426–449.

20. Bellman, R. (1954). The theory of dynamic programming. *Bull. Amer. Math. Soc*, **60**, 503–516.

21. Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, **4**(1), 238–252.

22. Berry, A. and Vamplew, P. (2006). An efficient approach to unbounded bi-objective archives—introducing the mak_tree algorithm. In *Proceedings of GECCO'06 – Genetic and Evolutionary Computation Conference*. 619–626.

23. Blum, J. (1954). Multidimensional stochastic approximation method. *Annals of Mathematical Statistics*, **25**, 737–744.

24. Box, G. (1957). Evolutionary operation: A method for incrasing industrial productivity. *Applied Statistics*, **6**, 81–101.

25. Box, G. and Behnken, D. (1960). Some new three-level designs for the study of quantitative variables. *Technometrics*, **6**, 247–262.

26. Box, G. and Draper, N. (1959). A basis for the selection of a response surface design. *Journal of the American Statistical Association*, **54**, 622–654.

27. Box, G. and Wilson, K. (1951). On the experimental attainment of optimaum conditions. *Journal of the Royal Statistical Society*, **13**(1), 1–45.

28. Choobineh, F., Mohebbi, E., and Khoo, H. (2006). A multi-objective tabu search for a single-machine scheduling problem with sequence-dependent setup times. *European Journal on Operational Research*, **175**(1), 318–337.

29. Ciprian, M., Clarich, A., Pediroda, V., and Poloni, C. (2006). Multi attribute design of airfoil under uncertainties by combining game theory and MCDM methods. In *Proceedings of the 18th International Conference on Multiple Criteria Decision Making*. [Online: accessed 25 June 2007].

30. Clarke, F. H. (1983). *Optimization and Nonsmooth Analysis*. Wiley, New York. Reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 5 in the series Classics in Applied Mathematics.

31. Coello Coello, C. (2005). Multiobjective structural optimization using a microgenetic algorithm. *Structural and Multidisciplinary Optimization*, **30**(5), 388–403.

32. Coello Coello, C. and Lamont, G., editors (2004). *Applications of Multi-Objective Evolutionary Algorithms, Advances in Natural Computation–Volume 1*. World Scientific, NJ, USA.

33. Collette, Y. and Siarry, P. (2004). *Multiobjective Optimization Principles and Case Studies*. Springer-Verlag, Berlin, Germany.

34. Conn, A. R., Gould, N. I., and Toint, P. L. (1991). A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, **28**(2), 545–572.

35. Cramer, E., Du, V., and Gablonsky, J. (2006). Multi-objective optimization for complex computer simulations. In *44th AIAA Aerospace Sciences Meeting and Exhibit*.

36. Cramer, E. J. (1998). Using approximate models for engineering design. In *7th Symposium on Multidisciplinary Analysis and Optimization*. AIAA Paper 1998-4716.

37. Das, I. and Dennis, Jr., J. E. (1997). A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimzation probems. *Structural Optimization*, **14**, 63–69.

38. Das, I. and Dennis, Jr., J. E. (1998). Normal-boundary intersections: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, **8**(3), 631–657.

39. Davis, C. (1954). Theory of positive linear dependence. *American Journal of Mathematics*, **76**(4), 733–746.

40. Day, R. O. and Lamont, G. B. (2005). An effective explicit building block MOEA, the MOMGA-IIa. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, volume 1. 17–24.

41. Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2001). Scalable test problems for evolutionary multi-objective optimization. TIK-Technical Report No. 112, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich Gloriastrasse 35., ETH-Zentrum, CH-8092, Zürich, Switzerland.

42. Dias, A. and de Vasconcelos, J. (2002). Multiobjective genetic algorithms applied to solve optimization problems. *IEEE Transactions on Magnetics*, **38**(2), 1133–1136.

43. Dias, L., Mousseau, V., Figueira, J., and Clímaco, J. (2002). An aggregation/disaggregation approach to obtain robust conclusions with electre tri. *European Journal of Operational Research*, **138**(2), 332–348.

44. Dolan, E., Lewis, R., and Torczon, V. (2003). On the local convergence of pattern search. *SIAM Journal on Optimization*, **14**(2), 567–583.

45. Dunlap, J. (2005). *On the Use of Surrogate Functions for Mixed Variable Optimization of Simulated Systems.* Master's thesis, Air Force Institute of Technology.

46. Ehrgott, M. (2005). *Multicriteria Optimization.* Springer, Berlin, Germany, second edition.

47. Ehrgott, M. and Gandibleux, X. (2004). Approximate solution methods for multiobjective combinatorial optimization. *Sociedad de Estadística e Investigación Operativa TOP*, **12**(1), 1–89.

48. Ermoliev, Y. and Wets, R., editors (1988). *Numerical Techniques for Stochastic Optimization.* Springer-Verlag, Berlin, Germany.

49. Ermoliev, Y. M. and Nurminski, V. (1980). *Stochastic Quasigradient Algorithms for Minimax Problems.* Academic Press, London, England, 275–285.

50. Fletcher, R. and Leyffer, S. (2002). Nonlinear programming without a penalty function. *Mathematical Programming*, **91**(2), 239–269.

51. Fonseca, C. and Fleming, P. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion, and generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California.* 416–423.

52. Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, **3**, 95–110.

53. Fu, M., Glover, F., and April, J. (2005). Simulation optimization: A review, new developments, and applications. In *Proceedings of the 2005 Winter Simulation Conference.*

54. Fu, Y. and Diwekar, U. (2004). An efficient sampling approach to multiobjective optimization. *Annals of Operations Research*, **132**, 109–134.

55. Gandibleux, X. and Freville, A. (2000). Tabu search based procedure for solving the 0-1 multi-objective knapsack problem: The two objective case. *Journal of Heuristics*, **6**(3), 361–383.

56. Georgopoulou, E., Sarafidis, Y., Mirasgedis, S., Zaimi, S., and Lalas, D. (2003). A multiple criteria decision-aid approach in defining national priorities for greenhouse gases emissions reduction in the energy sector. *European Journal of Operational Research*, **146**(1), 199–215.

57. Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, **13**(5), 533–549.

58. Glover, F., Laguna, M., and Marti, R. (2002). Scatter search and path relinking: Advances and applications.

59. Goldberg, D. E. (1989). *Genetic Algorithms: in Search, Optimization and Machine Learning*. Addison Wesly Publishing Company.

60. Goldsman, D. and Nelson, B. (1994). Ranking, selection, and multiple comparisions in computer simulation. In *Proceedings of the 1994 Winter Simulation Conference*.

61. Gosavi, A. (2003). *Simulation-Based Optimization, Parametric Optimization Techniques and Reinforcement Learning*. Kluwer Academic Publishers, Boston, MA, USA.

62. Granat, J. and Makowski, M. (2000). Interactive specification and analysis of aspiration-based preferences. *European Journal of Operational Research*, **122**, 469–485.

63. Hajela, P. and Lin, C. (1992). Genetic search strategies in multicriterion optimal design. *Structural Optimization*, **4**, 99–107.

64. Hansen, M. P. (1997). Tabu search for multiobjective optimization: Mots. In *Proceedings of the 13th International Conference on Multiple Criteria Decision Making*. [Online: accessed 28 August 2007].

65. Hapke, M., Jaszkiewicz, A., and Sowinski, R. (2000). Pareto simulated annealing for fuzzy multi-objective combinatorial optimization. *Journal of Heuristics*, **6**(3), 329–345.

66. Hochberg, Y. and Tamhane, A. (1987). *Multiple comparison Procedures*. John Wiley and Sons, Hoboken, NJ, USA.

67. Hooke, R. and Jeeves, T. (1961). Direct search solution of numerical and statistical problems. *Journal of the Association of Computing Machinery*, **8**, 212–229.

68. Horn, J., Nafpliotis, N., and Goldberg, D. (1994). A niched Pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation 1*. 82–87.

69. Howard, R. (1960). *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, USA.

70. Howell, M., Gordon, T., and Brandao, F. (2002). Genetic learning automata for function optimization. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, **32**(6), 804–815.

71. Humphrey, D. and Wilson, J. (1998). A revised simplex search procedurefor stochastic simulation response-surface optimization. In e. D. J. Medeiros, editor, *Proceedings of the 1998 Winter Simulation Conference*.

72. Humphrey, D. and Wilson, J. (2000). A revised simplex search procedure-for stochastic simulation response surface optimization. *INFORMS Journal on Computing*, **12**(4), 272–283.

73. Jagannathan, R. (1974). A sequential algorithm for a class of programming problems with nonlinear constraints. *Management Science*, **21**(1), 13–21.

74. Jahn, J. (1994). *Introduction to the Theory of Nonlinear Optimization*. Springer, Berlin.

75. Jin, R. and et al. (2001). Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Design Optimization*, **23**, 1–13.

76. Keifer, J. and Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, **23**, 462–466.

77. Kim, W., Grandhi, R. V., and Haney, M. (2005). Multi-objective evolutionary optimization method for thermal protection system design. In *46th Structures, Structural Dynamics, and Materials Conference*. AIAA Paper 2005-2311.

78. Kleeman, M. P. and Lamont, G. B. (2005). Solving the aircraft engine maintenance scheduling problem using a multiobjective evolutionary algorithm. In C. A. Coello Coello, A. H. Aguirre, and E. Zitzler, editors, *Evolutionary Multi-Criterion Optimization, Third International Conference, EMO 2005*. Springer-Verlag, Berlin, Germany, 782–796.

79. Knowles, J. and Corne, D. (1999). The Pareto archived evolution strategy: A new baseline for Pareto multiobjective optimisation. In *Proceedings of the 1999 Congress on Evolutionary Computation*. 98–105.

80. Kolda, T., Lewis, R., and Torczon, V. (2003). Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, **45**(3), 385–482.

81. Kwon, H., Park, S., and Lee, J. (2005). Transonic wing flutter simulation using Navier-Stokes and k-ω turbulent model. In *46th Structures, Structural Dynamics & Materials Conference*.

82. Langer, H., Puehlhofer, T., and Baier, H. A multi-objective evolutionary algorithm with integrated response surface functionaltities for configuration optimization with discrete variables. In *10th Multidisciplinary Analysis and Optimization Conference*. AIAA Paper 2004-4326.

83. Lee, L. H., Chew, E. P., and Teng, S. (2004). Optimal computing budget allocation for multi-objective simulation models. In J. S. S. R .G. Ingalls, M. D. Rossetti and B. A. Peters, editors, *Proceedings of the 2004 Winter Simulation Conference*.

84. Lee, L. H., Chew, E. P., and Teng, S. (2006). Integration of statistical selection with search mechanism for solving multi-objective simulation-optimization problems. In L. Perrone, F. Wieland, J. Liu, B. Lawson, D. Nicol, and R. Fujimoto, editors, *Proceedings of the 2006 Winter Simulation Conference*.

85. Lee, L. H., Chew, E. P., Teng, S., and Goldsman, D. (2005). Finding the nondominated Pareto set for multiobjective simulation models. Submitted to IIE Transactions.

86. Lee, L. H., Teng, S., Chew, E. P., Karimi, I., Lye, K. W., Lendermann, P., Chen, Y., and Koh, C. H. (2005). Application of multi-objective simulation-optimization techniques to inventory management problems. In F. B. A. M. E. Kuhl, N. M. Steiger and J. A. Joines, editors, *Proceedings of the 2005 Winter Simulation Conference*.

87. Lewis, R. and Torczon, V. (1999). Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, **9**(4), 1082–1099.

88. Lewis, R. and Torczon, V. (2000). Pattern search methods for linearly constrained minimization. *SIAM Journal on Optimization*, **10**(3), 917–941.

89. Lewis, R. and Torczon, V. (2001). A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization*, **12**(4), 1075–1089.

90. Lian, Y. and Liou, M.-S. (2005). Multi-objective optimization of a transonic compressor blade using evolutionary algorithm. In *46th Structures, Structural Dynamics, and Materials Conference*. AIAA Paper 2005-1816.

91. Lienard, S. and Lefevre, Y. (2005). Modeling and analysis of the deployment of a rolled inflatable beam using MSC-DYTRAN. In *46th Structures, Structural Dynamics, and Materials Conference*. AIAA Paper 2005-1968.

92. Lootsma, F., Athan, T., and Papalambros, P. (1995). Controlling the search for a compromise solution in multi-objective optimization. *Engineering Optimization*, **25**(1), 65–81.

93. Loukil, T., Teghem, J., and Tuyttens, D. (2005). Solving multi-objective production scheduling problems using metaheuristics. *European Journal on Operational Research*, **161**(1), 42–61.

94. Lucidi, S. and Piccialli, V. (2004). 371–.

95. Lucidi, S., Piccialli, V., and Sciandrone, M. (2005). An algorithm model for mixed variable programming. *SIAM Journal on Optimization*, **15**(4), 1057–1084.

96. Makowski, M. (1994). Methodology and a modular tool for multiple criteria analysis of lp models. Working Paper 94-102, International Institute for Applied Systems Analysis, Laxenburg, Austria.

97. Martí, R. (2006). Scatter search—wellsprings and challenges. *European Journal of Operational Research*, **169**(2), 351–358.

98. Martí, R., Laguna, M., and Glover, F. (2006). Principles of scatter search. *European Journal of Operational Research*, **169**(2), 359–372.

99. Martins, J. and Alonso, J. (2002). Complete configuration aero-structural optimization using a coupled sensitivity analysis method. In *9th Symposium on Multidisciplinary Analysis and Optimization*. AIAA Paper 2002-5402.

100. McKinnon, K. (1998). Convergence of the Nelder-Mead simplex method to a nonstationary point. *SIAM Journal on Optimization*, **9**(1), 148–158.

101. Miettinen, K. and Mäkelä, M. (2002). On scalarizing functions in multi-objective optimization. *OR Spectrum*, **24**, 193–213.

102. Miller, R. (2000). *Optimization Foundations and Applications*. John-Wiley & Sons, Inc., New York, NY.

103. Molina, J., Laguna, M., Marti, R., and Caballero, R. (2005). SSPMO: A scatter search procedure for non-linear multiobjective optimization.

104. Myer, R. H. and Montgomery, D. C. (1995). *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. John Wiley and Sons, Inc., New York, NY, USA.

105. Myers, R. H. and Carter, Jr., W. (1973). Response surface techniques for dual response systems. *Technometrics*, **15**(2), 301–317.

106. Nangia, U., Jain, N., and Wadhwa, C. (1997). Surrogate worth trade-off technique for multiobjective optimal power flows. *IEE Proceedings, Generation, Transmission and Distribution*, **144**(6), 547–553.

107. Nash, S. and Sofer, A. (1996). *Linear and Nonlinear Programming*. McGraw-Hill, New York, NY.

108. National Aeronautics and Space Administration (2007). NASA oral history project: NACA, national advisory committee for aeronautics. URL `http://www.jsc.nasa.gov/history/oral_histories/naca.htm`. [Online; accessed 31-July-2007].

109. Nelder, J. and Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, **7**(4), 308–313.

110. Nelson, B. L., Swann, J., Goldsman, D., and Song, W. (2001). Simple procedures for selecting the best simulated system when the number of alternatives is large. *Operations Research*, **49**(6), 950–963.

111. Norzari, A. and Morris, J. (1984). Application of an optimization procedure to steady-state simulation. In A. Sheppard et al., editor, *Proceedings of the 1984 Winter Simulation Conference*.

112. Oltean, M., Abraham, A., Grosan, C., and Köppen, M. (2005). Multiobjective optimization using adaptive Pareto archived evolution strategy. In *Proceedings of the 2005 5th International Conference on Intelligent Systems Design and Applications (ISDA'05)*. 93–100.

113. Pegden, C. and Gately, M. (1977). Decision optimization for GASP IV simulation models. In *Proceedings of the 1977 Winter Simulation Conference*.

114. Pegden, C. and Gately, M. (1980). A decision-optimization module for SLAM. *Simulation*, **34**(1), 18–25.

115. Pichitlamken, J. and Nelson, B. L. (2001). Selection-of-the-best procedures for optimization via simulation. In D. J. M. B. A. Peters, J. S. Smith and M. W. Rohrer, editors, *Proceedings of the 2001 Winter Simulation Conference*. 401–407.

116. Pohl, T., Grecksch, W., and Blaar, H. (2001). A parallel application of a quasi-gradient method in stochastic control theory. *Optimization*, **49**(1), 95–114.

117. Poloni, C. (1995). *Hybrid GA for Multi-Objective Aerodynamic Shape Optimisation*. Wiley and Sons, Chichester, 397–415.

118. Ramírez-Rosado, I. and Domínguez-Navarro, J. (2006). New multi-objective tabu search algorithm for fuzzy optimal planning of power distribution systems. *IIEEE Transactions on Power Systems*, **21**(1), 224–233.

119. Ray, T. and Liew, K. (2002). A swarm metaphor for multiobjective design optimization. *Engineering Optimization*, **34**(2), 141–153.

120. Reardon, B. J. (1997). Fuzzy logic vs. niched Pareto multiobjective genetic algorithm optimization: Part I: Schaffer's f2 problem. LA-UR-97-3675, Los Alamos National Laboratory, Los Alamos, New Mexico, September 1997.

121. Reardon, B. J. (1997). Fuzzy logic vs. niched Pareto multiobjective genetic algorithm optimization: Part II: A simplified born - mayer problem. LA-UR-97-3676, Los Alamos National Laboratory, Los Alamos, New Mexico, September 1997.

122. Rinott, Y. (1978). On two-stage selection procedures and related probability-inequalities. *Communications in Statistics*, (8).

123. Robbins, H. and Munro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, **22**, 400–407.

124. Rockafellar, R. T. (1980). Generalized directional derivatives and subgradients of nonconvex functions. *Canad. J. Math.*, **32**(2), 257–280.

125. Rodriguez, A. F., Keller, T. A., Lamont, G. B., and Nelson, T. R. (2005). Using a multiobjective evolutionary algorithm to develop a quantum cascade laser operating in the terahertz frequency range. Air Force Institute of Technology, Wright-Patterson AFB, OH 45433. [Online; accessed 20-February-2007].

126. Roskam, J. (1997). *Airplane Design, Part II: Preliminary Configuration Design and Integration of the Propulsion System*. Design, Analysis and Research Corporation (DARcorporation), Lawrence, KS.

127. Roskam, J. (2002). *Airplane Design, Part III: Layout Design of Cockpit, Fuselage, Wing, and Empannage: Cutaways and Inboard Profiles*. Design, Analysis and Research Corporation (DARcorporation), Lawrence, KS.

128. Sait, S. and Youssef, H. (1999). *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems*. IEEE Computer Society, Las Alamitos, CA, USA.

129. Sakawa, M., Kato, K., Sunada, H., and Shibano, T. (1997). Fuzzy programming for multiobjective 0-1 programming problems through revised genetic algorithms. *European Journal of Operational Research*, **97**(1), 149–158.

130. Salminen, P., Hokkanen, J., and Lahdelma, R. (1998). Comparing multicriteria methods in the context of environmental problems. *European Journal of Operational Research*, **104**(3), 485–496.

131. Sandia National Laboratories. Tabu search. URL `http://www.cs.sandia.gov/opt/survey/ts.html`.

132. Schaffer, J. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*. 93–100.

133. Serafini, P. (1992). Simulated annealing for multi objective optimization problems. In G. Tzeng, H. Wang, U. Wen, and P. Yu, editors, *Proceedings of the Tenth International Conference on Multiple Criteria Decision Making: Expand and Enrich the Domains of Thinking and Application*, volume 1. Springer-Verlag, 283–294.

134. Spall, J. (1993). *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley and Sons, Hoboken, NJ, USA.

135. Spendley, W., Hext, G., and Himsworth, F. (1962). Sequential application of simplex designs in optimization and evolutionary operation. *Technometrics*, **4**(4), 441–461.

136. Srinivas, N. and Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, **2**(3), 221–248.

137. Sriver, T. (2004). *Pattern Search Ranking and Selection Algorithms for Mixed-Variable Optimization of Stochastic Systems*. Ph.D. thesis, Air Force Institute of Technology.

138. Sriver, T. and Chrissis, J. (2004). Combined pattern search and ranking and selection for simulation optimization. In *Proceedings of the 2004 Winter Simulation Conference*.

139. Sriver, T., Chrissis, J., and Abramson, M. (2007). Pattern search ranking and selection algorithms for mixed variable simulation-based optimization. Air Force Institute of Technology, Wright-Patterson AFB, OH 45433. Preprint.

140. StatSoft inc. Nonlinear estimation. StatSoft Electronic Textbook. URL `http://www.statsoft.com/textbook/stnonlin.html`.

141. Suman, B. and Kumar, P. (2006). A survey of simulated annealing as a tool for single and multiple objective optimization. *Journal of the Operational Research Society*, **57**(10), 1143–1160.

142. Sweeney, D. J. and Murphy, R. A. (1979). A method of decomposition for integer programs. *Operations Research*, **27**(6), 1128–1141.

143. Swisher, J. (2000). A survey of simulation optimization techniques and procedures. In J. Joines, editor, *Proceedings of the 2000 Winter Simulation Conference*.

144. Tharaldson, D. (2006). *Optimization of a Multi-Echelon Repair System Via Generalized Pattern Search with Ranking and Selection: A Computational Study*. Master's thesis, Air Force Institute of Technology.

145. Tomick, J. et al. (1995). Sample size selection for improved Nelder-Mead performance. In C. Alexopoulous, et al., editor, *Proceedings of the 1995 Winter Simulation Conference*.

146. Torczon, V. (1997). On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, **7**(1), 1–25.

147. US Department of Energy. Computational science education project. URL `http://www.phy.ornl.gov/csep/mo/mo.html`.

148. Van Veldhuizen, D. (1999). *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. Ph.D. thesis, Air Force Institute of Technology.

149. Viennet, R. and Marc, I. (1996). Multicriteria optimization using a genetic algorithm for determining a Pareto set. *International Journal of Systems Science*, **27**(2), 255–260.

150. Walston, J. (2007). Search techniques for multi-objective optimization of mixed-variable systems having stochastic responses. Technical Report IR-07-014, International Institute for Applied Sytems Analysis, Schlossplatz 1, A-2361 Laxenburg, Austria.

151. Watkins, C. and Dayan, P. (1992). Q-learning. *Machine Learning*, **8**(3), 279–292.

152. Wauquiez, C. (1999). Potential flow around airfoils with boundary layer coupled one-way (PABLO). Freeware, used as lab material at the Royal Institute of Technology, Department of Aeronautics, Stockholm, Sweden.

153. Weisstein, E. W. (2007). Partial order. MathWorld–A Wolfram Web Resource. URL `http://mathworld.wolfram.com/PartialOrder.html`. Online: Accessed 4 September 2007.

154. Weisstein, E. W. (2007). Totally ordered set. MathWorld–A Wolfram Web Resource. URL `http://mathworld.wolfram.com/TotallyOrderedSet.html`. Online: Accessed 4 September 2007.

155. Wikipedia (2007). Bellman equation — Wikipedia, The Free Encyclopedia. URL `http://en.wikipedia.org/w/index.php?title=Bellman_equation`. [Online; accessed 5-February-2007].

156. Wikipedia (2007). Chord (aircraft) — Wikipedia, The Free Encyclopedia. URL `http://en.wikipedia.org/w/index.php?title=Chord_(aircraft)`. [Online; accessed 27-August-2007].

157. Wikipedia (2007). NACA airfoil — Wikipedia, The Free Encyclopedia. URL `http://en.wikipedia.org/w/index.php?title=NACA_airfoil`. [Online: accessed 2 July 2007].

158. Wikipedia (2007). Reynolds number — Wikipedia, The Free Encyclopedia. URL `http://en.wikipedia.org/w/index.php?title=Reynolds_number`. [Online; accessed 18-July-2007].

159. Wikipedia (2007). Subderivative — Wikipedia, The Free Encyclopedia. URL `http://en.wikipedia.org/w/index.php?title=Subderivative`. [Online; accessed 2-March-2007].

160. Wikipedia (2007). Total order — Wikipedia, The Free Encyclopedia. URL `http://en.wikipedia.org/w/index.php?title=Total_order`. [Online; accessed 4-September-2007].

161. Wikipedia (2007). Total relation — Wikipedia, The Free Encyclopedia. URL `http://en.wikipedia.org/w/index.php?title=Total_relation`. [Online; accessed 6-September-2007].

162. Winston, W. L. (1994). *Operations Research Applications and Algorithms*. Duxbury Press, Belmont, CA, USA, third edition.

163. Wu, J. and Azarm, S. (2001). Metrics for quality assessment of a multiobjective design optimization solution set. *Transactions of the ASME*, **123**(1), 18–25.

164. Zeng, X. and Liu, Z. (2005). A learning automata based algorithm for optimization of continuous complex functions. *Information Sciences*, **174**(3), 165–175.

165. Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, **3**(4), 257–271.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
09–2007	Doctoral Dissertation	Sep 2004 — Sep 2007

4. TITLE AND SUBTITLE

Search Techniques for Multi-Objective Optimization of Mixed Variable Systems having Stochastic Responses

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Jennifer G. Walston, Maj, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/DS/ENS/07S-06

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Dr Raymond M. Kolonay
AFRL/VASD
2130 Eighth Street, Suite 1, Bldg 45
Wright-Patterson AFB OH 45433-7542

Dr Margaret Goud Collins
U.S. Committee for IIASA
The National Academy of Sciences
500 Fifth Street
NW Washington, DC 20001

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

A research approach is presented for solving stochastic, multi-objective optimization problems. First, the class of mesh adaptive direct search (MADS) algorithms for nonlinearly constrained optimization is extended to mixed variable problems. The resulting algorithm, MV-MADS, is then extended to stochastic problems (MVMADS-RS), via a ranking and selection procedure. Finally, a two-stage method is developed that combines the generalized pattern search/ranking and selection (MGPS-RS) algorithms for single-objective, mixed variable, stochastic problems with a multi-objective approach that makes use of interactive techniques for the specification of aspiration and reservation levels, scalarization functions, and multi-objective ranking and selection. A convergence analysis for the general class of algorithms establishes almost sure convergence of an iteration subsequence to stationary points appropriately defined in the mixed-variable domain. Seven specific instances of the new algorithm are implemented and tested on 11 multi-objective test problems from the literature and an engineering design problem.

15. SUBJECT TERMS

multi-objective optimization, stochastic optimization, simulation-based optimization, mixed variable programming, interactive specification of aspiration/reservation levels, mesh adaptive direct search, pattern search

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr James W. Chrissis
U	U	U	UU	153	19b. TELEPHONE NUMBER *(include area code)* (937) 255–3636, ext 4606